Coordination of independent learners in cooperative Markov games.

Laëtitia Matignon, Guillaume J. Laurent, Nadine Le Fort-Piat

Institut FEMTO-ST

Université de Franche-Comté

Coordination of independent learners in cooperative Markov games

Laëtitia Matignon, Guillaume J. Laurent and Nadine Le Fort-Piat

Rapport technique / Technical Report

2009

Institut FEMTO-ST
UMR CNRS 6174 — UFC / ENSMM / UTBM
32 avenue de l'Observatoire — 25044 BESANCON Cedex FRANCE
Tél : (33 3) 81 85 39 99 — Fax : (33 3) 81 85 39 68
email : contact@femto-st.fr

Abstract

In the framework of fully cooperative multi-agent systems, independent agents learning by reinforcement must overcome several difficulties as the coordination or the impact of exploration. The study of these issues allows first to synthesize the characteristics of existing reinforcement learning decentralized methods for independent learners in cooperative Markov games. Then, given the difficulties encountered by these approaches, we focus on two main skills: optimistic agents, which manage the coordination in deterministic environments, and the detection of the stochasticity of a game. Indeed, the key difficulty in stochastic environment is to distinguish between various causes of noise. The SOoN algorithm is so introduced, standing for "Swing between Optimistic or Neutral", in which independent learners can adapt automatically to the environment stochasticity. Empirical results on various cooperative Markov games notably show that SOoN overcomes the main factors of non-coordination and is robust face to the exploration of other agents.

1 Introduction

Over the last decade, many approaches are concerned with the extension of RL to multi-agent systems (MAS) [1]. On the one hand, adopting a decentralized point of view with MAS offers several potential advantages as speed-up, scalability and robustness [2]. On the other hand, reinforcement learning (RL) methods do not need any *a priori* knowledge about the dynamics of the environment, which can be stochastic and non linear. An agent interacting with its environment tests different actions and learns a behavior by using a scalar reward signal called reinforcement as performance feedback. Modern RL methods rely on dynamic programming and have been studied extensively in single-agent framework [3]. By using RL in MAS, their advantages can be associated : autonomous and "simple" agents can learn to resolve in a decentralized way complex problems by adapting to them.

However, there are still many challenging issues in applying RL to MAS [4]. For instance, one difficulty is the loss of theoretical guarantees. Indeed convergence hypothesis of single-agent framework are no longer satisfied. Another matter is that the computation complexity of decentralized MAS often grows exponentially with the number of agents [5, 6, 7]. Finally, a fundamental difficulty faced by agents that work together is how to efficiently coordinate themselves [8, 9, 10]. MAS are qualified by a decentralized execution, so each agent takes individual decisions but all of the agents contribute globally to the system evolution. Learning in MAS is then more difficult since the selection of actions must take place in the presence of other learning agents. A study of some mis-coordination factors is proposed in this report.

Despite these difficulties, several successful applications of decentralized RL have been reported, like the control of a group of elevators [11] or adaptative load-balancing of parallel applications [12]. Air traffic flow management [13] and data transportation by a constellation of communication satellites [14] are other examples of real world applications of cooperative agents using RL algorithms. We have an interest in practical applications in robotics, where multiple cooperating robots are able to complete many tasks more quickly and reliably than one robot alone. Especially, the real world objective of our work is the decentralized control of a distributed micro-manipulation system based on autonomous distributed air-flow MEMS[1], called smart surface [15]. It consists of an actuators array on which an object is situated. The objective is for the hundreds of actuators to choose from a small set of actions over many trials so as to achieve a common goal.

We focus here on learning algorithm in cooperative MAS [16] which is a fitting model for such a real world scenario. Especially, in this report, we are interested in the coordination of

[1]Micro Electro Mechanical Systems.

adaptative and cooperative agents with the assumption of a full individual state observability. We consider independent learners (ILs) which were introduced in [17] as agents which don't know the actions taken by the other agents. The choice of ILs is particularly pertinent given our objective of robotic applications with numerous agents, where the assumption of joint action observability required for the joint action learners is hard to satisfy.

Our objective is to develop an easy-to-use, robust and decentralized RL algorithm for autonomous and independent learners in the framework of cooperative MAS. The chosen framework is cooperative Markov games. One of our contribution is to analyse what is at stake in the learning of ILs in this framework. Thanks to this study, we synthesize the characteristics of existing RL decentralized methods for ILs in cooperative Markov games. Then, we introduce a new algorithm named the SOoN algorithm, standing for "Swing between Optimistic or Neutral", in which ILs can adapt automatically to the environment stochasticity. We successfully develop this algorithm thanks to our analysis of previous algorithms and apply it within various matrix games and Markov games benchmarks. Its performance is also compared to other learning algorithms.

The report is organized as follows. In section 2, our theoretical framework is described. Learning in MAS has strong connections with game theory so a great part of the report concerns repeated matrix games. Matrix games study is a necessary first step toward the framework of Markov games. Indeed, it allows a better understanding of mis-coordination issues, presented in section 3 as one of major stakes of ILs in cooperative MAS. We overview related algorithms in section 4 and provide a uniform notation that has never been suggested and which emphasizes common points between some of these algorithms. This leads to a recursive version of the FMQ algorithm in the matrix game framework (section 6). It is successfully applied on matrix games which contain hard coordination challenges. Finally, in section 7, we present the SOoN algorithm in the general class of Markov games and show results demonstrating its robustness.

2 Theoretical framework

The studies of reinforcement learning algorithms in MAS are based on Markov games which define multi-agent multi-state models. They are the synthesis of two frameworks : Markov decision processes and matrix games. Markov decision processes are a single-agent multi-state model explored in the field of RL. Matrix games, on the other hand, are a multi-agent single-state model used in the field of game theory. These frameworks are independently considered in this section. In addition, some classifications and equilibrium concepts are outlined.

2.1 Markov Decision Processes

Definition 1 *A* Markov Decision Process (MDP) *[18, 19] is a tuple* $< S, A, T, R >$ *where :*

- S *is a finite set of states;*
- A *is a finite set of actions;*
- $T : S \times A \times S \mapsto [0; 1]$ *is a transition function that defines a probability distribution over next states.* $T(s, a, s')$ *is the probability of reaching* s' *when the action* a *is executed in* s*, also noted* $P(s'|s, a)$*,*
- $R : S \times A \times S \mapsto \mathbb{R}$ *is a reward function giving the immediate reward or* reinforcement *received under each transition.*

At each time step, the agent chooses an action a_t that leads the system from the state s_t to the new state s_{t+1} according to a transition probability $T(s_t, a_t, s_{t+1})$. Then it receives a reward $r_{t+1} = R(s_t, a_t, s_{t+1})$. Solving MDPs consists in finding a mapping from states to actions, called a *policy* $\pi : S \times A \mapsto [0; 1]$, which maximizes a criterion. $\pi(s, a)$ is the probability of choosing the action a in state s. Most of single agent RL algorithms are placed within this framework [20, 21, 3].

2.2 Matrix games

Game theory is commonly used to analyze the problem of multi-agent decision making, where a group of agents coexist in an environment and take simultaneous decisions. We first define the matrix game framework and the classification of these games, and then we examine two main equilibrium concepts.

2.2.1 Definition and strategy

Definition 2 *A* matrix game[2] *[22, 23] is a multiple agent, single state framework. It is defined as a tuple* $< m, A_1, ..., A_m, R_1, ..., R_m >$ *where :*

- m *is the number of players;*
- A_i *is the set of actions available to player* i *(and* $\boldsymbol{A} = A_1 \times ... \times A_m$ *is the joint action space);*
- $R_i : A \mapsto \mathbb{R}$ *is player* i*'s reward or* payoff *function.*

The goal of a learning agent in a matrix game is to learn a strategy that maximizes its reward. In matrix games, a strategy[3] $\pi_i : A_i \mapsto [0; 1]$ for an agent i specifies a probability distribution over actions, *i.e.* $\forall a_i \in A_i \ \pi_i(a_i) = P(a_i)$. $\boldsymbol{\pi}$ is the joint strategy for all of the agents and $\boldsymbol{\pi}_{-i}$ the joint strategy for all of the agents except agent i. We will use the notation $\langle \pi_i, \boldsymbol{\pi}_{-i} \rangle$ to refer to the joint strategy where agent i follows π_i while the other agents follow their policy from $\boldsymbol{\pi}_{-i}$. Similarly, $\boldsymbol{A}_{-i}$ is the set of actions for all of the agents except agent i. The set of all of the strategies for the agent i is noted $\Delta(A_i)$.

2.2.2 Type of matrix games

Matrix games can be classified according to the structure of their payoff function. If payoff functions sum is zero, the game is called *fully competitive game*[4]. If all agents have the same payoff function, *i.e.* $R_1 = ... = R_m = R$, the game is called *fully cooperative game*[5]. Finally, in *general-sum game*, the individual payoffs can be arbitrary.

In this paper, we focus on *fully cooperative game*. In these games, an action in the best interest of one agent is also in the best interest of all agents. Especially, we are interested in *repeated games* which consist of the repetition of the same matrix game by the same agents.

The game in table 1 depicts a team game between two agents which is graphically represented by a payoff matrix. The rows and columns correspond to the possible actions of respectively the first and second agent, while the entries contain the payoffs of the two agents for the corresponding joint action. In this example, each agent chooses between two actions a and b. Both agents receive a payoff of 0 when both selected actions of the agents differ, and a payoff of 1 or 2 when the actions are the same.

[2] also called *strategic game*.
[3] also called *policy*.
[4] also called *zero-sum game*.
[5] also called *team game* or *identical payoff game*.

Table 1: Fully cooperative game example.

		Agent 2	
		a	b
Agent 1	a	2	0
	b	0	1

2.2.3 Equilibria

First, we define the expected gain for an agent i as the expected reward given its strategy and the strategies of others.

Definition 3 *The expected gain for an agent i given a joint strategy $\boldsymbol{\pi} = \langle \pi_i, \boldsymbol{\pi}_{-i} \rangle$, noted $\mathsf{u}_{i,\boldsymbol{\pi}}$, is :*

$$\mathsf{u}_{i,\boldsymbol{\pi}} = E_{\boldsymbol{\pi}} \{R_i(a)\} = \sum_{a \in \boldsymbol{A}} \boldsymbol{\pi}(a) R_i(a). \tag{1}$$

From a game-theoretic point of view, two common concepts of equilibrium are used to define optimal solutions in games. The first one is called *Nash equilibrium* [24, 23].

Definition 4 *A joint strategy $\boldsymbol{\pi}^*$ defines a* Nash equilibrium *iff, for each agent i, there is :*

$$\forall \pi_i \in \Delta(A_i) \quad \mathsf{u}_{i,\langle \pi_i^*, \boldsymbol{\pi}_{-i}^* \rangle} \geq \mathsf{u}_{i,\langle \pi_i, \boldsymbol{\pi}_{-i}^* \rangle}. \tag{2}$$

Thus, *no agent can improve its payoff by unilaterally deviating from a Nash equilibrium.* A matrix game can have more than one Nash equilibrium. For example, the matrix game in table 1 has two Nash equilibria corresponding to the joint strategies where both agents select the same action. However, a Nash equilibrium is not always the best group solution. That's why the *Pareto optimality* has been defined.

Definition 5 *A joint strategy $\hat{\boldsymbol{\pi}}$* Pareto dominates *another joint strategy $\boldsymbol{\pi}$ iff :*

- *each agent i following $\hat{\pi}_i$ receives at least the same expected gain as with π_i and,*
- *at least one agent j following $\hat{\pi}_j$ receives a strictly higher expected gain than with π_j,*

that's to say formally :

$$\hat{\boldsymbol{\pi}} > \boldsymbol{\pi} \Leftrightarrow \forall i \quad \mathsf{u}_{i,\hat{\boldsymbol{\pi}}} \geq \mathsf{u}_{i,\boldsymbol{\pi}} \quad \textit{and} \quad \exists j \quad \mathsf{u}_{j,\hat{\boldsymbol{\pi}}} > \mathsf{u}_{j,\boldsymbol{\pi}}. \tag{3}$$

Definition 6 *If a joint strategy $\hat{\boldsymbol{\pi}}^*$ is not Pareto dominated by any other joint strategy, then $\hat{\boldsymbol{\pi}}^*$ is* Pareto optimal.

So a Pareto optimal solution is one in which no agent's expected gain can be improved upon without decreasing the expected gain of at least one other agent. There are many examples of general-sum games where a Pareto optimal solution is not a Nash equilibrium and vice-versa (for example, the prisoner's dilemma). However, in fully cooperative games, each Pareto optimal solution is also a Nash equilibrium by definition. This means that if a joint action provides one agent with maximal possible reward, it must also maximise the reward received by the other agents. For instance in the matrix game of the table 1, the joint strategy $\langle a, a \rangle$ defines a Pareto optimal Nash equilibrium.

2.3 Markov games

Markov games can be seen as the extension of matrix games to the multiple states framework. Specifically, each state of a Markov game can be viewed as a matrix game. Another point of view is to consider Markov games as an extension of MDP to multiple agents. Markov games were first examined in the field of game theory and more recently in the field of multi-agent RL.

2.3.1 Definition

Definition 7 *A* Markov game[6] *[25] is defined as a tuple* $< m, S, A_1, ..., A_m, T, R_1, ..., R_m >$ *where :*

- m *is the number of agents;*
- S *is a finite set of states;*
- A_i *is the set of actions available to the agent* i *(and* $A = A_1 \times ... \times A_m$ *the joint action space);*
- $T : S \times A \times S \mapsto [0; 1]$ *a transition function that defines transition probabilities between states;*
- $R_i : S \times A \mapsto \mathbb{R}$ *the reward function for agent* i*.*

In Markov game framework, all agents have access to the complete observable state s. The transition and reward functions are functions of the joint action. An individual policy $\pi_i : S \times A_i \mapsto [0; 1]$ for an agent i specifies a probability distribution over actions, *i.e.* $\forall a_i \in A_i \ P(a_i|s) = \pi_i(s, a_i)$. The set of all of the policies for the agent i is noted $\Delta(S, A_i)$. If for an agent i and for all states s, $\pi_i(s, a_i)$ is equal to 1 for a particular action a_i and 0 for the others, then the policy π_i is called a pure policy.

The same classification for matrix games can be used with Markov games. If all agents receive the same rewards, the Markov game is fully cooperative. It is then defined as an *identical payoff stochastic game* (IPSG)[7] [26].

2.3.2 Equilibria

As in matrix games, we can define for each state s the immediate expected gain for an agent i following the joint strategy π as :

$$\mathsf{u}_{i,\boldsymbol{\pi}}(s) = E_{\boldsymbol{\pi}}\left\{R_i(s, a)\right\} = \sum_{a \in \boldsymbol{A}} \boldsymbol{\pi}(s, a) R_i(s, a). \tag{4}$$

Thus, the long term expected gain from the state s for an agent i is, when all the agents follow π :

$$\mathsf{U}_{i,\boldsymbol{\pi}}(s) = E_{\boldsymbol{\pi}}\left\{\sum_{k=0}^{\infty} \gamma^k \mathsf{u}_{i,\boldsymbol{\pi}}(s)_{t+k+1} | s_t = s\right\} \tag{5}$$

where $\gamma \in [0; 1]$ is a discount factor.

The concepts of Nash equilibrium and Pareto optimality are also defined in Markov games.

[6] also called *stochastic game*.

[7] also called *Multi-agent Markov Decision Process* (MMDP).

Definition 8 *A joint policy $\boldsymbol{\pi}^*$ defines a Nash equilibrium in a Markov game iff, for each agent i, there is :*

$$\forall \pi_i \in \Delta(S, A_i)\ \forall s \in S \quad \mathsf{U}_{i,\langle \pi_i^*, \boldsymbol{\pi}_{-i}^* \rangle}(s) \geq \mathsf{U}_{i,\langle \pi_i, \boldsymbol{\pi}_{-i}^* \rangle}(s). \tag{6}$$

Definition 9 *A joint policy $\hat{\boldsymbol{\pi}}$* Pareto dominates *another joint policy $\boldsymbol{\pi}$ iff, in all states :*

- *each agent i following $\hat{\pi}_i$ receives at least the same expected gain as with π_i and,*
- *at least one agent j following $\hat{\pi}_j$ receives a strictly higher expected gain than with π_j,*

that's to say formally :

$$\hat{\boldsymbol{\pi}} > \boldsymbol{\pi} \Leftrightarrow \forall i,\ \forall s \in S \quad \mathsf{U}_{i,\hat{\boldsymbol{\pi}}}(s) \geq \mathsf{U}_{i,\boldsymbol{\pi}}(s) \quad et \quad \exists j \quad \mathsf{U}_{j,\hat{\boldsymbol{\pi}}}(s) > \mathsf{U}_{j,\boldsymbol{\pi}}(s). \tag{7}$$

Definition 10 *If a joint policy $\hat{\boldsymbol{\pi}}^*$ is not Pareto dominated by any other joint strategy, then $\hat{\boldsymbol{\pi}}^*$ is* Pareto optimal.

2.4 Conclusion

In this paper, we are interested in cooperative games, where we defined the objective as finding a Pareto-optimal Nash equilibrium, also called *optimal equilibrium*. Indeed, the policies which define Pareto-optimal Nash equilibria maximise the expected sum of the discounted rewards in the future for all agents. However, there can be several Pareto-optimal Nash equilibria so some coordination mechanisms on a single equilibrium are necessary.

3 The independent learners coordination problem

The main difficulty in the cooperative independent learners (ILs) framework is the *coordination* of ILs : how to make sure that all agents coherently choose their individual action such that the resulting joint action is optimal? We analyze this problem using the framework of fully cooperative matrix games that we have introduced in section 2.2. Each agent *independently* has to select an action from its action set. It receives a payoff based on the resulting *joint* action. Formally, the challenge in multi-agent RL is to guarantee that the individual optimal policies π_i^* define an optimal joint policy π^* [27], *i.e.* a Pareto-optimal Nash equilibrium.

The coordination is a complex problem which arises from the combined action of several factors. In large games, it is hard to identify these factors since they are often combined with other phenomena. So the classification proposed here is probably not exhaustive. It results from a study of matrix games with a small number of agents, where the identification and interpretation of some factors are more obvious. Of course, the exposed factors can be adapted to games with many agents and to Markov games. These factors make the analysis and development of coordination techniques in RL algorithms easier.

Fulda and Ventura [8] identify two main factors : *shadowed equilibrium* and *equilibria selection problems* that we formalized here. We also add two other factors : the *noise* in the game and the impact of the exploration of other agents on the learning of one agent.

3.1 Shadowed equilibrium

Definition 11 *An equilibrium defined by a strategy $\bar{\pi}$ is shadowed by a strategy $\hat{\pi}$ iff :*

$$\exists i \quad \exists \pi_i \quad \mathsf{u}_{\langle \pi_i, \bar{\boldsymbol{\pi}}_{-i} \rangle} < \min_{j,\pi_j} \mathsf{u}_{\langle \pi_j, \hat{\boldsymbol{\pi}}_{-j} \rangle} \tag{8}$$

Table 2: Simple cooperative matrix game.

		Agent 2		
		a	b	c
Agent 1	a	5	3	1
	b	4	1	1
	c	1	1	0

Table 3: The Climbing game and the Penalty game have been introduced in [28] and partially and fully stochastic variations of the Climbing game, proposed in [29]. In stochastic games, the probability of each reward is 50%.

(a) Climbing game

		Agent 2		
		a	b	c
Agent 1	a	11	-30	0
	b	-30	7	6
	c	0	0	5

(b) Partially stochastic Climbing game

		Agent 2		
		a	b	c
Agent 1	a	11	-30	0
	b	-30	14/0	6
	c	0	0	5

(c) Fully stochastic Climbing game

		Agent 2		
		a	b	c
Agent 1	a	10/12	5/-65	8/-8
	b	5/-65	14/0	12/0
	c	8/-8	12/0	10/0

(d) Penalty game ($k \leq 0$)

		Agent 2		
		a	b	c
Agent 1	a	10	0	k
	b	0	2	0
	c	k	0	10

A particular structure game could be a game with a shadowed Pareto optimal Nash equilibrium. For instance, in the Climbing game (table 3), the Pareto optimal Nash equilibrium $\langle a, a\rangle$ and the sub-optimal Nash equilibrium $\langle b, b\rangle$ are shadowed because penalties are associated to mis-coordination on $\langle a, b\rangle$ or $\langle b, a\rangle$. Moreover, there are no mis-coordination penalties associated with action c, potentially making it tempting for the agents. The joint action $\langle c, c\rangle$ is the only unshadowed equilibrium in the Climbing game. So $\langle c, c\rangle$ is potentially interesting for ILs.

3.2 Equilibria selection problems

The *equilibria selection problem* takes place whenever at least two agents are required to select between multiple Pareto optimal Nash equilibria. In this case, some agents must choose between several optimal individual policies but only some combinations of these optimal individual policies define an optimal equilibrium. So the equilibria selection problem can be defined as the coordination of individual policies so that the resulting joint policy is optimal.

Craig Boutilier introduced the set of *potentially individually optimal (PIO) actions* for agent i at state s as those actions in A_i that belong to at least one of the optimal joint actions for s [30]. An optimal joint action is a pure joint policy which defines a Pareto optimal Nash equilibrium. This set is denoted $PIO(i, s)$. State s is said to be *weakly dependent* for agent i if there is more than one PIO choice for i at s. We can define the equilibria selection problem with this notion.

Definition 12 *There is an equilibria selection problem at state s if at least two agents are required to select between multiple Pareto optimal Nash equilibria at this state,* i.e. *s is weakly dependent for at least two agents.*

The Penalty game (table 3d) is usually chosen to illustrate the equilibria selection problem. There are two Pareto optimal Nash equilibria : $\langle a, a\rangle$ and $\langle c, c\rangle$ and one sub-optimal Nash equilibrium $\langle b, b\rangle$. The set of potentially individually optimal actions for each agent is then $\{a, c\}$. So the Penalty game is weakly dependent for the two agents and agents confront an equilibria selection problem. Simply choosing an individual optimal action in the set PIO does not guarantee that the resulting joint action will be optimal since four joint actions are then possible ($\langle a, a\rangle$, $\langle c, c\rangle$, $\langle a, c\rangle$, $\langle c, a\rangle$) and only two of them are optimal.

In this game, k is usually chosen inferior to 0 so as to combine shadowed equilibria with equilibria selection issue. Indeed, the only unshadowed equilibrium in this game is the sub-optimal Nash equilibrium $\langle b, b\rangle$.

3.3 Noise in the environment

Agents in a MAS can be confronted with noise in their environment. Noise in the environment can be due to various phenomena : stochastic rewards, *e.g.* in the stochastic variations of the Climbing game (table 3d and c), unobservable factors in the state space, uncertainty, ... However, changing behaviors of the other agents can also be interpreted as noise by an IL. For instance in the partially stochastic Climbing game, an IL must distinguish if distinct rewards received for the action b are due to various behaviors of the other agent or to stochastic rewards. The agent must learn that the average reward for the joint action $\langle b, b\rangle$ is 7 and is lower than that of $\langle a, a\rangle$. Thus the difficulty in ILs games is to distinguish between noise due to the environment and noise due to other agents behaviors. This behavior is notably characterized by actions of exploration.

3.4 Exploration

In MAS as in single-agent case, each IL must choose actions of exploration or exploitation according to its decision strategy. The common action decision called *softmax* is based on agent's individual policy π_i computed thanks to a Boltzman distribution :

$$\pi_i(s,a) = \frac{e^{\frac{Q_i(s,a)}{\tau}}}{\sum_{u \in A_i} e^{\frac{Q_i(s,u)}{\tau}}} \tag{9}$$

where τ is the temperature parameter that decreases the amount of randomness as it approaches zero. The ϵ-*greedy* strategy is another common action selection method in which an agent follows its greedy policy[8] with probability $(1-\epsilon)$ (exploitation mode), and otherwise selects a uniformly random action with probability ϵ (exploration mode).

The exploration is a major stake in the reinforcement learning of ILs. Indeed, as stated before, an IL cannot detect the change of behaviors of the other agents. So the exploration of others can induce noise in received rewards of an IL.

So as to quantify the noise due to the exploration in a game, the concept of global exploration must be pointed out. The global exploration is the probability of having at least one agent which explores. It can be formulated with the individual exploration of each agent.

Property 1 *Let a n-agents system in which each agent explores according to a probability ϵ. Then the probability of having at least one agent which explores is $\psi = 1-(1-\epsilon)^n$. ψ is named global exploration.*

The exploration of an agent can have strong influences on the learned policy of an IL. In a game with an optimal shadowed equilibrium defined by a policy $\hat{\pi}$, let agent i follow its optimal individual policy $\hat{\pi}_i$. The exploration of others can then lead to penalties (equation 8). So the exploration of others has an influence on the learning of i's individual policy. Moreover, it can cause the "destruction" of the optimal policy of an IL. So *it is necessary that RL algorithms should be robust face to exploration.*

3.5 Conclusion

In this report, we focus our study on cooperative independent learners. In such a framework, the main issue is to manage the coordination of the learners. We have detailed four factors responsible of the non-coordination of agents. Solving these difficulties and overcoming mis-coordination factors is one the main objectives of RL algorithms in MAS. Another key parameter is the robustness face to exploration.

4 Related works in Reinforcement Learning

Various approaches have been proposed in the literature to prevent coordination problems in multi-independent-learners systems. In this section, related works dealing with RL algorithms are reviewed with an emphasis on algorithms dealing with Q-learning [31] and Q-learning variants for ILs. We propose a uniform notation, that has never been suggested for these algorithms, and which stresses common points between some of them. For each algorithms, the robustness face to mis-coordination factors and exploration is discussed.

[8] A greedy policy based on Q_i is to select in a state s the action a for which $Q_i(s,a)$ is highest.

Table 4: Percentage of trials which converged to the optimal joint action (averaged over 1000 trials). A trial consists of 3000 repetitions of the game. At the end of each trial, we determine if the greedy joint action is the optimal one. Results were obtained with softmax decision and various exploration strategies : stationary (τ) or GLIE ($\tau = \tau_{ini} \times \delta^t$ where t is the repetitions number).

Stationary strategy	$\tau = 1000$	$\tau = 200$	$\tau = 10$	$\tau = 1$
Penalty game ($k = -100$)	$62,6\%$	$65,3\%$	$62,8\%$	$65,1\%$

GLIE	$\tau_{ini} = 5000$				$\tau_{ini} = 500$	$\tau_{ini} = 100$
strategy	$\delta = 0.9$	$\delta = 0.99$	$\delta = 0.995$	$\delta = 0.997$	$\delta = 0.995$	$\delta = 0.995$
Penalty game ($k = -100$)	0%	$61,7\%$	84%	$96,6\%$	$85,3\%$	$84,8\%$

In the following methods, each IL builds its own Q_i-table whose size is independent of the agents number and linear in function of its own actions. The task is for the agents to independently choose one action with the goal of maximizing the expected sum of the discounted rewards in the future.

4.1 Decentralized Q-learning

Q-learning [31] is one of the most used algorithm is single-agent framework because of its simplicity and robustness. That's also why it was one of the first RL algorithm applied to multi-agent environments [17]. The update equation for the agent i is :

$$Q_i(s, a_i) \leftarrow (1 - \alpha)Q_i(s, a_i) + \alpha(r + \gamma \max_{u \in A_i} Q_i(s', u)) \tag{10}$$

where s' is the new state, a_i is the agent's chosen action, $r = R(s, \langle a_i, a_{-i} \rangle)$ the reward received, $Q_i(s, a_i)$ the value of the state-action for the agent i, $\alpha \in [0; 1]$ the learning rate and $\gamma \in [0; 1]$ the discount factor.

Decentralized Q-learning has been applied with success on some applications [32, 33, 34, 35]. However, this algorithm has a low robustness face to exploration. Most of works concerning decentralized Q-learning use a GLIE[9] strategy [36] so as to avoid concurrent exploration. It consists in avoiding the simultaneous exploration of agents, namely the concurrent exploration. The principle is to decrease the exploration frequency as the learning goes along so that each agent should find the best response to the others behaviors. This method is investigated in MAS [28, 37] and section 5.1.

Avoiding concurrent exploration can improved results of some algorithms by allowing ILs to overcome some mis-coordination factors. For instance, results of decentralized Q-learning can be improved as illustrated on table 4. Best results are obtained with a well chosen GLIE strategy. However, convergence then relies on the choice of decaying parameters. The use of a GLIE strategy does not ensure convergence to an optimal equilibrium [28] and the key difficulty of this method is that the convergence relies on the choice of decaying parameters. Moreover, setting GLIE parameters requires high skills on algorithm behavior and so it is difficult to apply in real applications.

[9] Greedy in the limit with infinite exploration.

Algorithm 1: Distributed Q-learning for agent i [10]

begin

Initialization :

forall $a \in A_i$ *and* $s \in S$ **do**

$Q_{i,max}(s,a) \leftarrow 0$, $\pi_i(s,a)$ arbitrarily

$s \leftarrow$ initial state

while s *is not an absorbing state* **do**

From s select a according to the ϵ-greedy selection method based on π_i

Apply a and observe reward r and next state s'

$q \leftarrow r + \gamma \max_{u \in A_i} Q_{i,max}(s',u)$

if $q > Q_{i,max}(s,a)$ **then**

$Q_{i,max}(s,a) \leftarrow q$ ▷ `optimistic update`

if $Q_{i,max}(s, \arg\max_{u \in A_i} \pi_i(s,u)) \neq \max_{u \in A_i} Q_{i,max}(s,u)$ **then**

Select a random action $a_{max} \in \arg\max_{u \in A_i} Q_{i,max}(s,u)$

$\forall b \in A_i \quad \pi_i(s,b) \leftarrow \begin{cases} 1 & \text{if } b = a_{max} \\ 0 & \text{else} \end{cases}$ ▷ `equilibria selection mechanism`

$s \leftarrow s'$

end

4.2 Distributed Q-learning

To get round the issue of *shadowed equilibria* with ILs, Lauer & Riedmiller [38] introduced "optimistic independent agents". Such agents neglect in their update the penalties which are often due to a non-coordination of agents. For instance the evaluation of an action in matrix games is the maximum reward received. In the case of multiple optimal joint actions in a single state, an additional procedure for coordination is used to solve the equilibria selection problem. This equilibria selection mechanism is a social convention [30] which places constraints on the possible action choices of the agents. The central idea is to update the current policy π_i only if an improvement in the evaluation values ($Q_{i,max}$) happens. This mechanism presents the other interest to allow simultaneously the learning of individual policies and exploration. The individual policies can't be "destroyed" by exploration since when one of the optimal joint policies is tried, it is nevermore modified.

Distributed Q-learning associated with this equilibria selection method is in algorithm 1. It addresses two factors that can cause mis-coordination : shadowed equilibria are prevented through the optimistic update and the equilibria selection problem is solved thanks to the equilibria selection method. *It is proved that this algorithm finds optimal policies in deterministic environments for cooperative Markov games.* Moreover, distributed Q-learning is robust face to exploration. However this approach does generally not converge in stochastic environments.

4.3 Variable learning rate algorithms

To minimize the effect of *shadowed equilibria* with ILs, some authors propose to minimize the effect that the learning of other agents has on a given agent's own learning by using *a variable learning rate*. For instance some algorithms based on gradient ascent learners use the Win or Learn Fast (WoLF) heuristic [39] and its variants PD-WoLF [40] or GIGA-WoLF [41]. Others are based on Q-learning, *e.g.* the hysteretic Q-learning [42] in which two learning rates α and

β are used for the increase and decrease rates of Q_i-values. The update equation for the agent i is :

$$\delta \leftarrow r + \gamma \max_{u \in A_i} Q_i(s', u) \tag{11}$$

$$Q_i(s,a) \leftarrow \begin{cases} (1-\alpha)Q_i(s,a) + \alpha\delta & \text{if } \delta \geq Q_i(s,a) \\ (1-\beta)Q_i(s,a) + \beta\delta & \text{else} \end{cases} \tag{12}$$

The idea is that agents should not be altogether blind to penalties at the risk of staying in sub-optimal equilibrium or mis-coordinating on the same optimal joint equilibrium. But they are chiefly optimistic to reduce oscillations in the learned policy ($\alpha > \beta$).

4.4 Lenient Multiagent Reinforcement Learning

Panait *et al.* [43, 44] are interested in varying the degree of optimism of the agents as the game is repeated. Indeed, being optimistic may be useful at early stages of learning to identify promising actions. In case of lenient learners, agents are exploring at the beginning so most selected actions are poor choices and ignoring penalties is then justified. Nevertheless, it may lead to an overestimation of actions, especially in stochastic domains where rewards are noisy. And once agents have explored, it becomes interesting to achieve accurate estimation of actions. So the agents are initially lenient (or optimistic) and the degree of lenience concerning an action decreases as the action is often selected. The main drawbacks of this method are that a large number of parameters must be set and that it is only proposed for matrix games.

4.5 Frequency Maximum Q-value (FMQ)

Kapetanakis & Kudenko [29, 45] bias the probability of choosing an action with the frequency of receiving the maximum reward for that action. In their algorithm, the evaluation of an action E_i is the Q_i-value added to an *heuristic value*, taking into account how often an action produces its maximum corresponding reward. The evaluation of an action a is defined as :

$$E_i(a) = Q_i(a) + c \times F_i(a) \times Q_{i,max}(a) \tag{13}$$

where $Q_{i,max}(a)$ is the maximum reward received so far for choosing action a, $F_i(a)$ is the frequency of receiving the maximum reward corresponding to an action, named *occurrence frequency*, and c is a weight which controls the importance of the FMQ heuristic in the evaluation. *The Frequency Maximum Q value (FMQ)* algorithm is described in the algorithm 2. $C_i(a)$ holds the number of times the agent has chosen the action a in the game and $C_{i,Q_{max}}(a)$ the number of times that the maximum reward has been received as a result of playing a. Although FMQ algorithm is only proposed for matrix games, it shows interesting results on partially stochastic games.

Thanks to the uniform notation that we have proposed in this section, it is obvious that the FMQ has common points with the distributed Q-Learning. Indeed, the same $Q_{i,max}$ table is computed in both algorithms.

Kapetanakis & Kudenko [29] use with FMQ a softmax decision method and an exponentially decaying temperature function :

$$\tau_k = \tau_{max} e^{-\delta k} + \tau_\infty \tag{14}$$

[10] Rewards are supposed to be non-negative.
[11] Rewards are supposed to be non-negative.

Algorithm 2: FMQ for Matrix Game for agent i [11]

begin

Initialization : $\forall a \in A_i,\ Q_i(a) \leftarrow 0,\ Q_{i,max}(a) \leftarrow 0,\ C_i(a) \leftarrow 0$
$C_{i,Q_{max}}(a) \leftarrow 0,\ F_i(a) \leftarrow 1,\ E_i(a) \leftarrow 0,\ \pi_i$ arbitrarily

repeat

Select a following the policy π_i
Apply a and observe reward r
$C_i(a) \leftarrow C_i(a) + 1$ ▷ `action occurrence counter`
$Q_i(a) \leftarrow (1-\alpha)Q_i(a) + \alpha r$
if $r > Q_{i,max}(a)$ **then**
$Q_{i,max}(a) \leftarrow r$ ▷ `optimistic update`
$C_{i,Q_{max}}(a) \leftarrow 1$ ▷ `maximal reward occurrence counter`
else if $r = Q_{i,max}(a)$ **then**
$C_{i,Q_{max}}(a) \leftarrow C_{i,Q_{max}}(a) + 1$
$F_i(a) \leftarrow \frac{C_{i,Q_{max}}(a)}{C_i(a)}$ ▷ `occurrence frequency`
$E_i(a) \leftarrow Q_i(a) + c \times F_i(a) \times Q_{i,max}(a)$ ▷ `heuristic`
$\forall b \in A_i,\ \pi_i(b) \leftarrow \frac{e^{\frac{E_i(b)}{\tau}}}{\sum_{u \in A_i} e^{\frac{E_i(u)}{\tau}}}$

until *the end of the repetitions*

end

Table 5: Percentage of trials which converged to the optimal joint action (averaged over 500 trials). A trial consists of 5000 repetitions of the game. We set $\alpha = 0.1$ and $c = 10$. At the end of each trial, we determine if the greedy joint action is the optimal one.

			FMQ
Exploration strategy	GLIE	$\tau = 499e^{-0,006t} + 1$	100%
		$\tau = 100e^{-0,006t} + 1$	59%
		$\tau = 499e^{-0,03t} + 1$	86%
		$\tau = 100 \times 0,997^t$	70%
	Stationary	$\tau = 20$	23%

where k is the number of repetitions of the game so far, δ controls the rate of the exponential decay, τ_{max} and τ_∞ set the values of the temperature at the beginning and at the end of the repetitions. However, using such a temperature function requires to choose in an appropriate manner all the decaying parameters. Results in table 5 illustrate the difficulty to choose these parameters; changing just one setting in these parameters can turn a successful experiment into an unsuccessful one. So *FMQ has a low robustness face to exploration.*

4.6 Experimental results

We compare the performance of some of these algorithms in the cooperative matrix games presented in table 2 and 3. A trial consists of 3000 repetitions of the game. At the end of each trial, we determine if the greedy joint action is the optimal one. Results were obtained with the *best chosen and tuned action selection strategy* in order to achieve the best results of convergence (see Appendix A). Table 6 brings together the percentage of trials converging to the optimal joint action according to the algorithm and the type of cooperative matrix.

Table 6: Percentage of trials which converged to the optimal joint action (averaged over 500 trials). Entries marked with "NT" indicate it has not been tested.

	Decentralized Q-learning	Distributed Q-learning	Hysteretic Q-learning	FMQ
Simple cooperative game	100%	100%	100%	100%
Climbing game	3%	100%	100%	100%
Penalty game ($k = -100$)	96%	100%	100%	100%
Partially stochastic Climbing game	NT	7%	82%	98%
Fully stochastic Climbing game	NT	NT	0%	21%

First, it is interesting to notice that with an appropriate exploration strategy, all algorithms find the optimal joint action in the simple cooperative game. Concerning the distributed Q-learning, the convergence is managed with a stationary strategy in deterministic games only. The decentralized Q-learning succeeds in selecting the equilibrium in the Penalty game thanks to the GLIE strategy which plays an implicit role of equilibria selection mechanism. However, it could be more difficult to overcome shadowed equilibria as in the Climbing game. Penalties induce decentralized Q-learning agents to converge towards unshadowed equilibria so it fails. GLIE strategy does not ensure convergence to an optimal equilibrium. In deterministic matrix games, hysteretic Q-learning and FMQ achieve the convergence with a GLIE strategy. However, FMQ heuristic show the best results in partially and fully noisy games. *The advantage of the FMQ is to be able to make the distinction between the noise due to the non-coordination of agents and the noise due to the environment in weak noise games.*

4.7 Conclusion

Table 7 sums up which mis-coordination factors each algorithm is able to overcome and its robustness face to exploration. We specify that a GLIE strategy is difficult to set and requires high skills about the system and the algorithm. In view of this synthesis, two methods can be outlined. On the one hand, distributed Q-learning has been proved to converge in deterministic Markov games, and in this framework, it overcomes all factors and is robust face to exploration. But it ceases to be effective when stochastic rewards are introduced. On the other hand, FMQ heuristic is partially effective in non-deterministic matrix games and it could be an interesting way to detect the stochasticity causes of a game. Moreover, thanks to our uniform notation, common points between FMQ and distributed Q-learning have been shown, and especially they compute the same $Q_{i,max}$ tables. So it would be desirable to have an algorithm that improves the results of coordination techniques by combining these both methods. The objective is an algorithm robust face to exploration and able to overcome mis-coordination factors in deterministic and stochastic Markov games.

With the view to combine these two methods, we are first interested in improving FMQ heuristic in cooperative matrix games (section 5). Indeed FMQ requires a GLIE strategy and is weakly robust face to exploration. These improvements lead to a recursive FMQ designed for matrix games, robust face to exploration and gathering FMQ heuristic and optimistic indepen-

Table 7: Characteristics of RL algorithms for independent learners in cooperative games. Entries marked with "NT" indicate it has not been tested.

	Matrix Games	Markov Games	Equilibria Selection	Shadowed Equilibria	Stochastic environment	Robust face to exploration
Decentralized Q-Learning [31]	✓	✓	✓ with GLIE		✓ partially	low
Distributed Q-Learning [38]	✓	✓	✓	✓		total
Lenient learners [43]	✓		NT	NT	NT	low
Hysteretic Q-Learning [42]	✓	✓	✓ with GLIE	✓	✓ partially	low
WoLF PHC [39]	✓	✓	✓		NT	good
FMQ [29]	✓		✓ with GLIE	✓	✓ partially	low

dent agents (section 6). Then, we extend this algorithm to Markov game framework (section 7).

5 Frequency Maximum Q-value study

In this section, we are interested in improving FMQ heuristic in cooperative matrix games. Indeed, one of its difficulty is its low robustness face to exploration. Especially, there remain questions towards understanding exactly how the exploration strategy influences the convergence. The convergence also relies on the choice of the weight parameter c. Moreover, the evaluation of the action proposed in the original FMQ is not relevant. In this section, we investigate these issues in details and suggest some improvements. *The objective is to achieve an algorithm more robust to the exploration and to get rid of the choice of the weight parameter c thanks to a novel evaluation of the action.*

5.1 Improving the robustness face to the exploration strategy

5.1.1 Link between convergence and exploration

The low robustness of FMQ face to exploration has been illustrated in §4.5. To study these link between convergence and exploration in the FMQ, an exponentially decaying temperature is plotted in function of the number of repetitions of the game in figure1. The other curves are the average rewards and their dispersion received by FMQ agents using this temperature function and the softmax strategy in the Climbing game. Two phases can be identified. During a first phase of *exploration*, the average rewards remain constant; the agents choose all possible joint actions. Then, during the exponential decay of τ, the agents learn to coordinate until the temperature reaches some lower limit where agents are following their greedy policy. This is a phase of *coordination*.

Both of these phases of exploration and coordination are necessary for the FMQ to converge. If these phases are wrongly set or if a stationary strategy is used, FMQ success rate highly

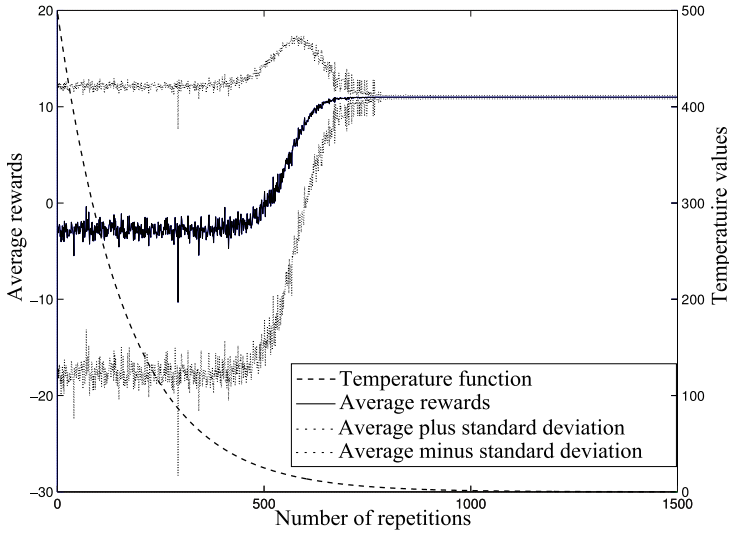


Figure 1: Average rewards received in the Climbing game by FMQ agents ($c = 10$) with $\tau = e^{0.006 \times k} \times 499 + 1$.

decreases (table 5). So FMQ has a low robustness face to exploration, *i.e.* the learning of a policy cannot be simultaneous to the exploration.

5.1.2 Instability of the frequency face to exploration

The link between convergence and exploration in the FMQ is due to an instability of the occurrence frequency F face to exploration. We illustrate this phenomenon on the Climbing game. We choose to study the instability of the occurrence frequency of the agent 1 for its action a, *i.e.* $F_1(a)$, face to the exploration of the other agent. a is the optimal individual action so $F_1(a)$ must tend toward 1.

Agents are supposed to be at the beginning of a learning phase. So the frequency $F_1(a)$ is initialized to 1. We plot the evolution of $F_1(a)$ according to the action choice of the other agent[12] (Fig. 2 dotted line). Actions chosen by the agents are represented on the diagram[13]. From beginning to end, the agent 1 only chooses action a and we study the instability of its frequency $F_1(a)$ face to the exploration of the other agent.

At the beginning, the joint action $\langle a, c \rangle$ is chosen many times. Agent 1 always receives the same reward for the action a. So $C_1(a)$ is high. Then, when the optimal joint action $\langle a, a \rangle$ is chosen for the first time (because of an exploration step), a new maximum reward is received for the action a and :

- the maximum reward received so far is modified $Q_{1,max}(a) \leftarrow 11$,
- the number of times that the maximum reward has been received as a result of playing a is reset $C_{1,Q_{max}}(a) \leftarrow 1$,
- the number of times the agent has chosen the action $C_1(a)$ is incremented.

[12] An agent has 3 actions a, b and c.
[13] Agents are following a "manual" policy.

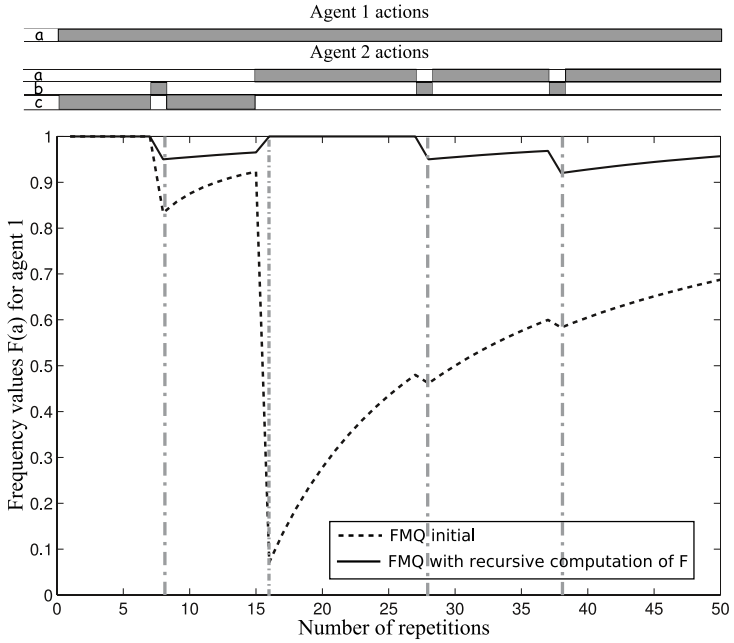


Figure 2: Frequency values of agent 1 *vs.* the number of repetitions of the game and according FMQ algorithm. Actions chosen by the agents are in the diagram above the curve.

The frequency value, updated according to $F_1(a) \leftarrow \frac{C_{Q_{1,max}}}{C_1(a)} = \frac{1}{C_1(a)}$, drops at the 15th repetition (dotted line). Consequently, the action evaluation $E_1(a)$ might then be smaller than other action evaluations and the action a might be chosen only in exploration step. The convergence is not effective and many exploration steps are necessary so that the frequency $F_1(a)$ may increase enough for the coordination on the optimal joint action. Moreover, the later the optimal joint action is played, the more important is the drop in frequency values, and the higher is the number of exploration steps necessary for the coordination. So the optimal joint action must be chosen as sooner as possible. That's why the first phase of exploration is so important in the FMQ.

5.1.3 Recursive computation of the frequency

In order to obtain an algorithm more robust to the exploration strategy, the counter $C_1(a)$ must be reset to 1 when a new maximum reward is received for the action a. The frequency $F_1(a) \leftarrow \frac{1}{C_1(a)}$ is then also reset to 1 as at the fifteenth repetition in figure 2 (solid line). Thus, the frequency $F_1(a)$ is maintained around 1 and will only decrease in case of mis-coordination because of an exploration step of one agent or in case of noisy reward. The frequency is then more stable face to exploration.

Using incremental counters as C or $C_{Q_{max}}$ can also be a dependence factor on the oldness. The oldness is that the increase or decrease rate of the frequency depends on the number of repetitions after which an action of exploration or exploitation is played. So we introduce a recursive computation of the frequency :

$$F(a) \leftarrow \begin{cases} (1-\alpha_f)F(a) + \alpha_f & \text{if } r = Q_{max}(a) \\ (1-\alpha_f)F(a) & \text{else} \end{cases} \tag{15}$$

Table 8: Percentage of trials which converged to the optimal joint action (averaged over 500 trials). A trial consists of 5000 repetitions of the game. We set $\alpha = 0.1$, $\alpha_f = 0.05$ and $c = 10$. At the end of each trial, we determine if the greedy joint action is the optimal one.

			FMQ	Modified FMQ
Exploration strategy	GLIE	$\tau = 499e^{-0,006^t} + 1$	100%	100%
		$\tau = 100e^{-0,006^t} + 1$	59%	100%
		$\tau = 499e^{-0,03^t} + 1$	86%	100%
		$\tau = 100 \times 0,997^t$	70%	100%
	Stationary	$\tau = 20$	23%	100%

where α_f is the learning rate of the frequency. It is obvious in figure 2 that the frequency is now more robust to the exploration (solid line). The choice of α_f will be discussed in §6.

5.1.4 Results

To check the robustness of modified FMQ with a reset and a recursive computation of the frequency, we choose the Climbing game. FMQ and its modified version are compared. Results are given in table 8. Modified FMQ converges with all tested strategies. Especially, a stationary strategy can be chosen. Using a constant exploration rate requires less parameters to set and *is less inclined to instability due to the exploration*. The only care concerns a sufficient number of repetitions of the game according to the constant exploration rate to ensure a complete exploration of the action set.

5.2 A new heuristic for the evaluation of the action

Kapetanakis & Kudenko [29] introduce a weight c to control the importance of the FMQ heuristic in the evaluation of an action (equation 13). But the convergence relies on the value of this parameter [29]. Moreover, Kapetanakis & Kudenko do not justify their choice of an action evaluation which is the sum of Q_{max} values, weighted by c, F, and Q values. What are the theoretical justifications for using such a sum as an action evaluation? We propose another heuristic which has the advantage of avoiding the choice of any parameters and which is bound to converge in deterministic games.

First, we remind that both FMQ and distributed Q-learning compute the same Q_{max} tables. This function returns in most cases an overestimation of the action evaluation, except in deterministic games where it is the *exact maximal value of an action*, as shown by Lauer & Riedmiller [38]. FMQ also uses Q-values which are an underestimation of the average rewards, since they include poor values due to mis-coordination. So real action values are between Q_{max} and Q values. If Q_{max} values are used to evaluate the actions in deterministic matrix game, the agents are ensured to converge if they use an equilibria selection mechanism. If the environment is stochastic, there is not any fitting algorithm so a solution could be to use Q values as action evaluations.

So if we can detect the stochasticity of the game, it is possible to evaluate actions with Q_{max} is the game is deterministic or with Q values otherwise. To evaluate the stochasticity, we use the occurrence frequency F of receiving the maximum reward corresponding to an action. Indeed, if the game is deterministic and if the agents manage the coordination, the frequency tends to 1. So we propose to use a linear interpolation to evaluate an action a :

$$E(a) = [1 - F(a)]Q(a) + F(a)Q_{max}(a). \tag{16}$$

Thus, actions evaluation is close to optimistic values if the game is deterministic. Otherwise, it fluctuates between Q_{max} and Q values according to the coordination of the agents. Indeed, when a new maximum reward for an action is received, the agent is first optimistic concerning this action. Then, if the reward is noisy, the frequency decreases and the agent becomes less optimistic and chooses its action according to Q-values. Thus this evaluation fluctuates between optimistic evaluation of distributed agents and mean evaluation of decentralized Q-learning agents according to the detection of the stochasticity.

This action evaluation is a heuristic, so it can be modified or improved. For instance, an other option could be to apply an "all or nothing" evaluation instead of a linear interpolation. Then, a bound would be used for the frequency such that the agents would be optimistic ($E = Q_{max}$) if the frequency was upper than this bound, and they would be neutral ($E = Q$) if the frequency was lower than this bound. But such a heuristic requires to determine this bound.

5.3 Conclusion

Two main limitations of FMQ have been studied : its low robustness face to exploration and face to the weight parameter. Then improvements have been suggested. First, a recursive computation of the frequency is proposed to improve the robustness. Second, a new heuristic based on a linear interpolation between Q_{max} and Q values is introduced. Thus, actions evaluation fluctuates between optimistic and mean evaluations according to the stochasticity of the game, which is estimated with the occurrence frequency. This method is set out in detail in the next section where these modifications are applied to the FMQ.

6 Recursive FMQ for cooperative matrix games

Previous modifications are incorporated to modify FMQ. The new algorithm, named recursive FMQ, is described in algorithm 3. Q_{max} function is updated as in distributed Q-learning. In parallel are updated Q-values. These functions are used as bounds in the action evaluation. The evaluation is a linear interpolation heuristic based on the occurrence frequency. Actions evaluations fluctuate between Q_{max} and Q values according to the stochasticity. The occurrence frequency is recursively computed thanks to a learning parameter α_f and reset when a new maximum reward is received for an action. Thus if the game is deterministic and if the agents manage to coordinate on an optimal joint action, frequencies of individual optimal actions will be close to 1. Agents will be then optimistic. Otherwise, evaluations fluctuate between optimistic and mean values. The equilibria selection mechanism of [38] is used, with a random choice among actions that successfully maximizes the evaluation E.

The principle of this algorithm is then closest in spirit to lenient learners [43] (section 4.4) given that the degree of optimism of the agents can change. But concerning lenient learners, the degree of optimism inevitably decreases, although here, *if the agents manage to coordinate in a deterministic environment, they stay optimistic.* So they are bound to converge to the optimal joint action in deterministic environment, given that they follow the distributed Q-learning.

One of our stated goals was to obtain a more robust algorithm face to exploration. With this intention we have suggested a recursive computation of the frequency and removed the weight parameter c which influenced the convergence. Then a new parameter α_f, named the

[14] Rewards are supposed to be non-negative.

Algorithm 3: Recursive FMQ for Matrix Game for agent i [14]

begin
  Initialization :
  forall $a \in A_i$ **do**
    $Q_i(a) \leftarrow 0$, $Q_{i,max}(a) \leftarrow 0$, $F_i(a) \leftarrow 1$, $E_i(a) \leftarrow 0$, $\pi_i(a)$ arbitrarily
  repeat
    Select a according to the ϵ-greedy selection method based on π_i
    Apply a and observe reward r
    $Q_i(a) \leftarrow (1-\alpha)Q_i(a) + \alpha r$
    if $r > Q_{i,max}(a)$ **then**
      $Q_{i,max}(a) \leftarrow r$ ▷ `optimistic update`
      $F_i(a) \leftarrow 1$ ▷ `frequency reset`
    else if $r = Q_{i,max}(a)$ **then**
      $F_i(a) \leftarrow (1-\alpha_f)F_i(a) + \alpha_f$ ▷ `recursive computation of the frequency`
    else
      $F_i(a) \leftarrow (1-\alpha_f)F_i(a)$
    $E_i(a) \leftarrow [1-F_i(a)]Q_i(a) + F_i(a)Q_{i,max}(a)$ ▷ `linear interpolation heuristic`
    if $E_i(\arg\max_{u\in A_i} \pi_i(u)) \neq \max_{u\in A_i} E_i(u)$ **then**
      Select a random action $a_{max} \in \arg\max_{u\in A_i} E_i(u)$
      $\forall b \in A_i \quad \pi_i(b) \leftarrow \begin{cases} 1 & \text{if } b = a_{max} \\ 0 & \text{else} \end{cases}$ ▷ `equilibria selection`
  until *the end of the repetitions*
end

learning rate parameter of the recursive frequency, has been introduced. First, the benefit of removing the parameter c is that the new evaluation, based on a linear interpolation, has been justified and is more relevant. Second, the choice of α_f can be stated so as to obtain a robust frequency. This will be the topic of the next paragraph.

6.1 Setting of the learning rate of the frequency α_f

The frequency role is to detect the stochasticity in the game. So it must be able to set apart noise due to the non-coordination from noise due to the environment. So it is necessary that the frequency of one agent should not be sensitive to the exploration of others. Indeed, as stated in §3.4, the exploration of others can be considered as noise by an IL. The discrimination between various noise causes is all the more complex as agents number increases. Indeed, many agents imply that the noise due to their exploration increases. Thus the impact of many agents in a system can start to look like stochasticity. So we must make sure of the robustness face to the exploration of the frequency. α_f choice is important and must be specified in function of global exploration.

α_f sets the increase and decrease rates of the recursive frequency. *With $\alpha_f = 0$, recursive FMQ is the same as distributed Q-learning.* The higher is α_f, the more sensitive is the frequency to the exploration of others. Indeed, let two agents be coordinated in a deterministic game, the agents' occurrence frequencies for the optimal individual actions are close to 1. If one explores, the received reward is smaller than the maximum reward. Then, for the greedy agent, the

Table 9: Advocated choices for the learning rate of the frequency α_f *vs.* global exploration ψ.

Global exploration	Learning rate of recursive frequency
$\psi = 0.1$	$\alpha_f = 0.01$
$\psi = 0.15$	$\alpha_f = 0.001$
$\psi = 0.2$	$\alpha_f = 0.0001$

Table 10: Percentage of trials which converged to the optimal joint action (averaged over 500 trials). A trial consists of 5000 repetitions of the game. The exploration strategy is stationary. We set $\alpha = 0.1$, $\alpha_f = 0.01$ and $\epsilon = 0.05$. At the end of each trial, we determine if the greedy joint action is optimal (x%) or sub-optimal (y%). Results are noted x%[y%].

	Simple game	Penalty game ($k = -100$)	Climbing game		
			deterministic	partially stochastic	fully stochastic
recursive FMQ	100%	100%	100%	100%	56% [3%]

frequency decreases according to $(1 - \alpha_f)$. So if α_f is too much high, occurrence frequencies can be "destroyed" by exploration of others. *α_f must be chosen all the smaller as the global exploration is high.*

Advocated choices for the learning rate of the frequency are given in table 9 *vs.* global exploration. These choices arise from experimentations and lead to a robust frequency face to exploration. We advise against choosing a GLIE strategy because the choice of α_f would then be complex[15]. This is in accordance with our previous warning about using such strategies, given their setting difficulties and their influence over the convergence. However, a stationary strategy is well fitted. In this case, a sufficient exploration must be set on the one hand for agents to find the optimal joint policy, and on the other hand for frequencies to converge toward their right values. Yet the global exploration must be restricted to avoid too much noise. Typically we suggest the value 0.2 as maximum bound for the global exploration.

6.2 Results on cooperative matrix games

We try recursive FMQ algorithm on various matrix games with 2 and more agents. These games bring together some mis-coordination factors. A key parameter of these tests is the exploration strategy and the repetitions number of the game per trial. As stated previously, we use a stationary strategy with recursive FMQ, since it is easier to set and the choice of α_f is less complex (table 9). Nevertheless, it leads with many agents to a large number of repetitions. Indeed, on the one hand, a sufficient exploration must be realized for agents to find optimal equilibria. But on the other hand, the global exploration is limited to avoid too much noise.The number of joint actions is exponential with the number of agents. So the number of repetitions per trial is growing with the number of agents.

6.2.1 2 agents games

We test the performance of recursive FMQ on cooperative matrix games presented in table 2 and 3. Results are given in table 10. Recursive FMQ is used with a stationary strategy (ϵ-greedy). It succeeds in all deterministic games and in partially stochastic Climbing game. Yet

[15]GLIE is not necessary anymore.

Table 11: Percentage of trials which converged to the optimal joint action in Penalty game with $n > 2$ agents (averaged over 500 trials). Chosen parameters are specified in Appendix B. At the end of each trial, we determine if the greedy joint action is optimal (x%) or sub-optimal (y%). Results are noted x%[y%].

		Decentralized Q-learning		Distributed Q-learning		WoLF PHC		Recursive FMQ	
$n =$ 3	deterministic	100%	[0%]	100%	[0%]	100%	[0%]	100%	[0%]
	stochastic	98%	[2%]	0%	[100%]	87%	[13%]	99%	[1%]
$n =$ 4	deterministic	100%	[0%]	100%	[0%]	100%	[0%]	91%	[9%]
	stochastic	6%	[94%]	0%	[100%]	0%	[100%]	92%	[8%]
$n =$ 5	deterministic	100%	[0%]	100%	[0%]	100%	[0%]	97%	[3%]
	stochastic	10%	[90%]	0%	[100%]	1%	[99%]	94%	[6%]

only half trials converge to the optimal in the fully stochastic Climbing game. So results are equivalent to original FMQ but the main interest of recursive FMQ is *its robustness face to exploration* thanks to a relevant choice of the frequency learning rate in function of the global exploration.

6.2.2 n agents games with $n > 2$

The role of the frequency is to distinguish noise from stochasticity versus noise from the other agents. In order to check that the frequency sets apart noise sources, we test recursive FMQ on large matrix games. Indeed, when there are many agents, it is more difficult to distinguish noise due to the environment from noise due to the others exploration.

We use a n agents Penalty game detailed in appendix B. Recursive FMQ is compared with other RL algorithms. Results are given in table 11. Decentralized Q-learning succeeds in all deterministic games but with a well-chosen GLIE strategy. In stochastic games, its performances greatly reduce with the number of agents. However it is difficult to blame a wrong choice of the GLIE parameters or the lack of robustness face to noisy environments of the algorithm. Similarly, WoLF-PHC algorithm fails as soon as rewards are noisy. But we stress that in stochastic games, these algorithms converge toward sub-optimal equilibria.

Concerning distributed Q-learning, results comply with theoretical guarantee [38]. It succeeds in all deterministic games but fails in stochastic games.

Lastly, best results are achieved with recursive FMQ. It succeeds more than nine times out of ten with all games. In particular, it outperforms decentralized Q-learning in stochastic games with $n > 3$ agents. *So the linear interpolation is a good measure of real actions values and the frequency manages the distinction between noise from stochasticity and noise from the other agents.*

6.3 Conclusion

This section leads to a recursive version of the FMQ algorithm, which is more robust to the choice of the exploration strategy. A means of decoupling various noise sources in a matrix games has been studied. The recursive frequency sets apart noise due to others exploration from noise due to stochasticity. The choice of frequency parameter has been stated versus global exploration so as to obtain a robust frequency.

Recursive FMQ is effective with a constant exploration rate and is bound to converge in deterministic matrix game. It also outperforms the difficulties of partially stochastic environ-

Algorithm 4: Straightforward extension of recursive FMQ to Markov Games for i[16]

begin
  Initialization :
  forall $a \in A_i$ *and* $s \in S$ **do**
    $Q_i(s,a) \leftarrow 0$, $Q_{i,max}(s,a) \leftarrow 0$, $F_i(s,a) \leftarrow 1$, $E_i(s,a) \leftarrow 0$, $\pi_i(s,a)$ arbitrarily
  $s \leftarrow$ initial state
  while s *is not an absorbing state* **do**
    From s select a according to the ϵ-greedy selection method based on π_i
    Apply a and observe reward r and next state s'
    $Q_i(s,a) \leftarrow (1-\alpha)Q_i(s,a) + \alpha(r + \gamma \max_{u \in A_i} Q_i(s',u))$
    $q \leftarrow r + \gamma \max_{u \in A_i} Q_{i,max}(s',u)$
    if $q > Q_{i,max}(s,a)$ **then**
      $Q_{i,max}(s,a) \leftarrow q$ ▷ `optimistic update`
      $F_i(s,a) \leftarrow 1$ ▷ `occurrence frequency reset`
    else if $q = Q_{i,max}(s,a)$ **then**
      $F_i(s,a) \leftarrow (1-\alpha_f)F_i(s,a) + \alpha_f$ ▷ `recursive frequency`
    else
      $F_i(s,a) \leftarrow (1-\alpha_f)F_i(s,a)$
    $E_i(s,a) \leftarrow [1 - F_i(s,a)]Q_i(s,a) + F_i(s,a)Q_{i,max}(s,a)$ ▷ `linear interpolation heuristic`
    if $E_i(s, \arg\max_{u \in A_i} \pi_i(s,u)) \neq \max_{u \in A_i} E_i(s,u)$ **then**
      Select a random action $a_{max} \in \arg\max_{u \in A_i} E_i(s,u)$
      $\forall b \in A_i \quad \pi(s,b) \leftarrow \begin{cases} 1 & \text{if } b = a_{max} \\ 0 & \text{else} \end{cases}$ ▷ `equilibria selection`
    $s \leftarrow s'$
end

ments. *The result is an easy-to-use and robust algorithm able to solve hard repeated matrix games.* But an extension to Markov games is necessary.

7 The Swing between Optimistic or Neutral algorithm

In this section, we plan to design an algorithm for ILs in cooperative Markov games. We first propose a straightforward extension to Markov games of our previous recursive FMQ algorithm. But this extension presents few limits and some modifications are required to manage the coordination. We introduce a fitting frequency for multi-state framework. The Swing between Optimistic or Neutral (SOoN) algorithm is then proposed. We apply the SOoN algorithm to various multi-state benchmarks.

7.1 Straightforward extension to Markov games

The straightforward extension of recursive FMQ to Markov Games is in algorithm 4. This extension is obvious except the occurrence frequency $F_i(s,a)$ computation.

The frequency $F_i(a)$ computed in matrix games is the probability of receiving the maximum

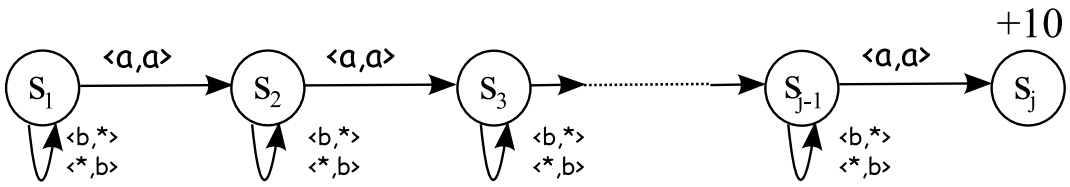


Figure 3: Two agents cooperative Markov game

reward corresponding to an action a :

$$F_i(a) = P_r\left\{r_{t+1} = Q_{i,max}(a_t) | a_t = a\right\}. \tag{17}$$

In Markov games, the frequency is defined as following and named myopic frequency..

Definition 13 *When action a is taken at state s, the myopic frequency $F_i(s,a)$ of agent i is the probability that the immediate transition should return a reward and a state whose evaluation, according to a greedy policy on $Q_{i,max}$, is equal to the expected maximum value of $Q_{i,max}(s,a)$,* i.e. *:*

$$F_i(s,a) = P_r\left\{r_{t+1} + \gamma \max_{b \in A_i} Q_{i,max}(s_{t+1}, b) = Q_{i,max}(s_t, a_t) | s_t = s, a_t = a\right\}. \tag{18}$$

In the straightforward FMQ extension, following functions are defined for each individual agent i :

- the value function Q_i updated as in decentralized Q-learning,
- the value function $Q_{i,max}$ updated as in distributed Q-learning,
- the myopic frequency F_i computed according to its definition 13,
- the evaluation function E_i based on a linear interpolation between Q_i and $Q_{i,max}$ values,
- the policy π_i updated according to an equilibria selection mechanism based on a random choice among actions maximizing the evaluation function E_i.

The myopic frequency is an immediate measure taking only the frequency in the current state into account. But is this myopic frequency enough for multi-state games?

7.2 Limitations of the myopic frequency

The straightforward extension can raise issues in some cases. To explain these limitations, we perform experiments on the cooperative Markov game given in figure 3, detailed in appendix C. This benchmark is difficult because agents must coordinate in each state. Especially, we study the development of the myopic frequency.

Optimal values of $Q_{i,max}$ for each agent i, at state s_k, are :

$$\begin{cases} Q^*_{i,max}(s_k, a) = 10 \times \gamma^{j-k-1} \\ Q^*_{i,max}(s_k, b) = 10 \times \gamma^{j-k} \end{cases}$$

From the point of view of one agent, we study two cases.

[16] Rewards are supposed to be non-negative.

- The agent i plays the action b at state s_k, so whatever the other agent plays, the next state s' is s_k and for each agent i :

$$q_i = \gamma \max_{u=1..2} Q_{i,max}(s_k, u) = 10 \times \gamma^{j-k} = Q^*_{i,max}(s_k, b) \tag{19}$$

  So from the viewpoint of an agent, the action b is reliable and the occurrence frequency of receiving the $Q_{i,max}$ for the action b, namely $F_i(s_k, b)$, increases each time action b is played at s by i. Moreover, it never decreases so $F_i(s_k, b)$ converges to 1.

- The agent plays the action a at state s_k. If the other agent also plays a, they coordinate and the next state s' is s_{k+1} :

$$q_i = \gamma \max_{u=1..2} Q_{i,max}(s_{k+1}, u) = 10 \times \gamma^{j-k-1} = Q^*_{i,max}(s_k, a) \tag{20}$$

  If the other agent plays b, they mis-coordinate and the next state s' is s_k :

$$q_i = \gamma \max_{u=1..2} Q_{i,max}(s_k, u) = 10 \times \gamma^{j-k} < Q^*_{i,max}(s_k, a) \tag{21}$$

  So $F_i(s_k, a)$ decreases each time the other agent plays b.

On the one hand, $F_i(s, b)$ increases each time action b is played at state s by agent i and never decreases. On the other hand, $F_i(s, a)$ decreases each time agents mis-coordinate at s. Consequently, from the viewpoint of one agent, the action b is safer than a. $Q^*_{i,max}(s, a)$ is close to $Q^*_{i,max}(s, b)$ because $Q^*_{i,max}(s, b) = \gamma Q^*_{i,max}(s, a)$. The evaluation $E(s, b)$ can be higher than $E(s, a)$. So the exploitation strategy of the agents is to choose the joint action $< b, b >$ and so, to remain on the spot. Its failure to coordinate is illustrated in results presented in figure 4 at page 28 (dotted line). The number of steps per trial quickly increases since agents remain on the spot and mis-coordinate. At the beginning of the learning, agents have a random behavior so the number of steps per trial is low.

This example illustrates a limitation in the straightforward extension of the recursive FMQ to Markov games due to the myopic frequency. This immediate measure only concerns the frequency in the current state, which is inadequate in multi-state games. The frequency must be globally considered by taking frequencies of future states into account. Indeed, in a given state, the myopic frequency of an action could be high, but this action could lead to a state where the optimal action has a low myopic frequency. In other words, an action could appear interesting in the current step according to its immediate frequency but could be a bad choice considering the future.

Using a myopic frequency could mislead the algorithm. So it could be relevant to use a long term or farsighted frequency.

7.3 Farsighted frequency

Definition 14 *When action a is played at state s, the* farsighted frequency $G_i(s, a)$ *for agent i is the probability that* all *future transitions return rewards and states whose evaluations, according to a greedy policy on $Q_{i,max}$, are equal to the expected maximum value $Q_{i,max}$ for these states,* i.e. *:*

$$G_i(s, a) = P_r^{\pi_{max}} \Big\{ r_{t+1} + \gamma \max_{b \in A_i} Q_{i,max}(s_{t+1}, b) = Q_{i,max}(s_t, a_t) \wedge$$
$$r_{t+2} + \gamma \max_{b \in A_i} Q_{i,max}(s_{t+2}, b) = Q_{i,max}(s_{t+1}, a_{t+1}) \wedge \ldots | s_t = s, a_t = a \Big\} \tag{22}$$

The question is how to estimate this farsighted frequency?

We make the assumption that precedent events are independent, *i.e.* the occurrence of one event makes it neither more nor less probable that the other occurs. So the statement above is equivalent to :

$$G_i(s,a) = P_r^{\pi_{max}}\left\{r_{t+1} + \gamma \max_{b\in A_i} Q_{i,max}(s_{t+1},b) = Q_{i,max}(s_t,a_t) | s_t = s, a_t = a\right\} \times$$
$$P_r^{\pi_{max}}\left\{r_{t+2} + \gamma \max_{b\in A_i} Q_{i,max}(s_{t+2},b) = Q_{i,max}(s_{t+1},a_{t+1}) \wedge \ldots | s_t = s, a_t = a\right\}. \quad (23)$$

According to the definition 13, the first term of the previous equation is equal to $F_i(s,a)$. And thanks to the total probability theorem, the second term is :

$$P_r^{\pi_{max}}\left\{r_{t+2} + \gamma \max_{b\in A_i} Q_{i,max}(s_{t+2},b) = Q_{i,max}(s_{t+1},a_{t+1}) \wedge \ldots | s_t = s, a_t = a\right\} =$$
$$\sum_{s'\in S} P_r^{\pi_{max}}\left\{r_{t+2} + \gamma \max_{b\in A_i} Q_{i,max}(s_{t+2},b) = Q_{i,max}(s_{t+1},a_{t+1}) \wedge \ldots | s_t = s, a_t = a, s_{t+1} = s'\right\} \times$$
$$P_r\left\{s_{t+1} = s' | s_t = s, a_t = a\right\}. \quad (24)$$

The π_{max} policy is following so $a_{t+1} \in \arg\max_{b\in A_i} Q_{i,max}(s_{t+1},b)$ and :

$$G_i(s,a) \quad = \quad F_i(s,a) \sum_{s'\in S} P_r\left\{s_{t+1} = s' | s_t = s, a_t = a\right\} G_i(s', \arg\max_{b\in A_i} Q_{i,max}(s_{t+1},b)) \quad (25)$$

If $P_{ss'}^a$ is the probability of each next possible state s' given a state s and an action a, we obtain :

$$G_i(s,a) = F_i(s,a) \sum_{s'\in S} P_{ss'}^a G_i(s', \arg\max_{u\in A_i} Q_{i,max}(s',u)) \quad (26)$$

As $P_{ss'}^a$ is unknown, we suggest to evaluate the farsighted frequency by following a temporal difference recursive computation :

$$G_i(s,a) \leftarrow (1-\alpha_g) G_i(s,a) + \alpha_g F_i(s,a) \max_{v\in \arg\max_{u\in A_i} Q_{i,max}(s',u)} G_i(s',v) \quad (27)$$

where $\alpha_g \in [0;1]$ is the learning rate of the farsighted frequency. The improvement supplied to the farsighted frequency is the product of the current state-action couple myopic frequency with the myopic frequency of the optimal action in the next state. Thus, if an action has at current state a high myopic frequency, but can lead to a state where the optimal action has a low farsighted frequency, then, this action will have a low farsighted frequency.

We could draw connections between the farsighted frequency general idea and bonus propagation for exploration [46]. Bonus propagation leads to a full exploration of the environment thanks to a mechanism of retro-propagation concerning local measures of exploration bonuses. In the case of the farsighted frequency, values of myopic frequencies are spread : for each visited state, the agent "looks forward" future states to take into account values of frequencies of optimal actions in these states.

Algorithm 5: SOoN algorithm for Markov Game for agent i[17]

begin

Initialization :

forall $a \in A_i$ *and* $s \in S$ **do**

$Q_i(s,a) \leftarrow 0$, $Q_{i,max}(s,a) \leftarrow 0$, $F_i(s,a) \leftarrow 1$, $G_i(s,a) \leftarrow 1$, $E_i(s,a) \leftarrow 0$, $\pi_i(s,a)$ arbitrarily

$s \leftarrow$ initial state

while s *is not an absorbing state* **do**

From s select a according to the ϵ-greedy selection method based on π_i

Apply a and observe reward r and next state s'

$Q_i(s,a) \leftarrow (1-\alpha)Q_i(s,a) + \alpha(r + \gamma \max_{u \in A_i} Q_i(s',u))$

$q \leftarrow r + \gamma \max_{u \in A_i} Q_{i,max}(s',u)$

if $q > Q_{i,max}(s,a)$ **then**

$Q_{i,max}(s,a) \leftarrow q$ ▷ `optimistic update`

$F_i(s,a) \leftarrow 1$ ▷ `myopic frequency reset`

$G_i(s,a) \leftarrow 1$ ▷ `farsighted frequency reset`

else if $q = Q_{i,max}(s,a)$ **then**

$F_i(s,a) \leftarrow (1-\alpha_f)F_i(s,a) + \alpha_f$ ▷ `frecursive myopic frequency`

else

$F_i(s,a) \leftarrow (1-\alpha_f)F_i(s,a)$

▷ `farsighted frequency update`

$G_i(s,a) \leftarrow (1-\alpha_g)G_i(s,a) + \alpha_g F_i(s,a) \max_{\substack{v \in \arg\max_{u \in A_i} Q_{i,max}(s',u)}} G_i(s',v)$

▷ `linear interpolation heuristic`

$E_i(s,a) \leftarrow [1 - G_i(s,a)]Q_i(s,a) + G_i(s,a)Q_{i,max}(s,a)$

if $E_i(s, \arg\max_{u \in A_i} \pi_i(s,u)) \neq \max_{u \in A_i} E_i(s,u)$ **then**

Select a random action $a_{max} \in \arg\max_{u \in A_i} E_i(s,u)$

$\forall b \in A_i \quad \pi_i(s,b) \leftarrow \begin{cases} 1 & \text{if } b = a_{max} \\ 0 & \text{else} \end{cases}$ ▷ `equilibria selection`

$s \leftarrow s'$

end

7.4 Swing between Optimistic or Neutral algorithm

We use the farsighted frequency in a new algorithm, named Swing between Optimistic or Neutral (SOoN) and detailed in algorithm 5. Same functions as those defined at §7.1 are used. The novelty is the state-action evaluation based on the farsighted frequency G. The evaluation of a state-action couple sways from distributed Q_{max} values to decentralized Q values according to G values and so according to a detection of stochasticity in the game. In deterministic Markov games, the farsighted frequency tends toward 1 and thus, the SOoN algorithm is close to the distributed Q-learning and is insured to converge towards optimal joint actions. Otherwise, in stochastic Markov games, the farsighted frequency is between 0 and 1 and the SOoN algorithm swings between optimistic or neutral evaluations.

We apply the SOoN algorithm to the cooperative Markov game of the figure 3 where the coordination of the two agents is difficult. Results are given in figure 4 (solid line). It is obvious

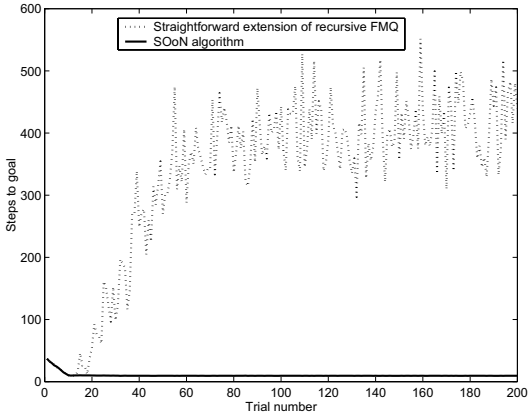


Figure 4: Two agents cooperative Markov game experiments with 10 states averaged over 200 runs ($\alpha = 0.1$, $\gamma = 0.9$, $\alpha_f = 0.05$, $\alpha_g = 0.3$, $\epsilon = 0.05$). Steps to goal *vs.* trial number.

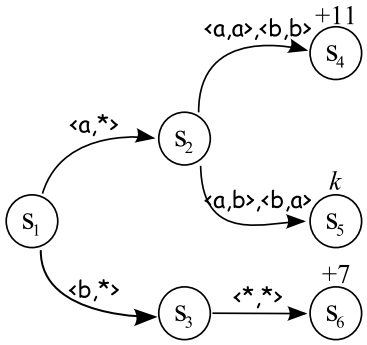


Figure 5: Boutilier's coordination game.

that the agents then manage the coordination after 15 trials thanks to the use of the farsighted frequency.

7.5 Experimentations on Markov game benchmarks

We now show results of applying the SOoN algorithm to a number of different Markov games. We compare the performance of decentralized Q-learning, distributed Q-learning, WoLF-PHC, hysteretic Q-learning and SOoN. The domains include various levels of difficulty according to the agents number (2 and more), the state observability (full or incomplete), and mis-coordination factors (shadowed equilibria, equilibria selection, deterministic or stochastic environments). For each benchmarks, we state what is at stake in the learning of ILs. Moreover, and in a concern to be fair, all algorithms used ϵ-greedy selection method with a stationary strategy and global exploration of $\psi = 0.1$. It allows to check the robustness face to the exploration of each method.

7.5.1 Boutilier's coordination Markov games

The Boutilier's coordination game with two agents was introduced in [47] and is shown in figure 5. This multi-state benchmark is interesting because all mis-coordination factors can

[17] Rewards are supposed to be non-negative.

Table 12: Percentage of cycles which converged to the optimal joint action (averaged over 50 cycles). We set $\alpha = 0.1$, $\gamma = 0.9$, $\delta_{lose} = 0.006$, $\delta_{win} = 0.003$, $\alpha_f = 0.01$, $\alpha_g = 0.03$, $\epsilon = 0.05$. $\beta = 0.01$ in deterministic games and $\beta = 0.05$ in stochastic games. At the end of each cycle, we determine if the greedy joint action is optimal.

	Boutilier's coordination game			
	deterministic		stochastic	
	$k = 0$	$k = -100$	$k = 0$	$k = -100$
Decentralized Q-learning	100%	6%	100%	12%
Distributed Q-learning	100%	100%	0%	0%
WoLF-PHC	100%	36%	100%	40%
Hysteretic Q-learning	100%	100%	100%	30%
SOoN	100%	100%	100%	96%

easily be integrated and combined. At each state, each agent has a choice of two actions. The transitions on the diagram are marked by a pair of corresponding actions, denoting player 1's move and player 2's move respectively. " * " is a wild card representing any action. States that yield a reward are marked with the respective value. All other states yield a reward of 0. A coordination issue applies in the second state of the game. Both agents should then agree on the same action. Moreover, there are two optimal equilibria so the agents are face to an equilibria selection problems and must coordinate on the same. Mis-coordination in the second state produces a penalty k. So the potential of choosing the action a for the first agent in the state s_1 is *shadowed* by the penalty k associated to mis-coordination. We also propose a partially stochastic version of the Boutilier's game with a random reward (14 or 0) received in state s_6 with equal probability instead of 7.

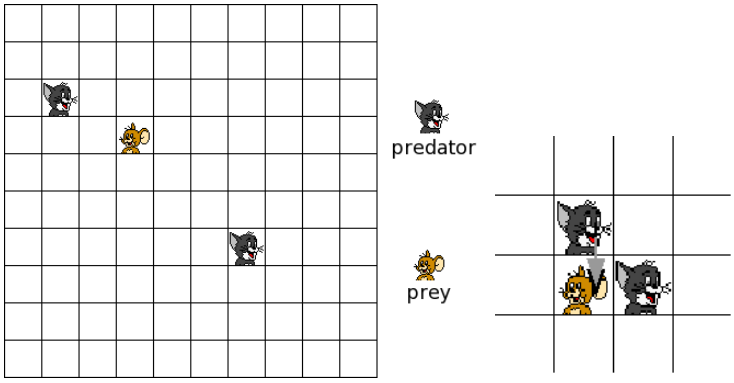
A trials ends when agents reach one of the absorbing states (s_4, s_5 or s_6). One cycle consists of 50000 learning trials. At the end of each cycle, we determine if the greedy joint action is optimal. Results are in table 12.

In deterministic game with $k = 0$, all algorithms manage the coordination. Similarly, in the stochastic game with $k = 0$, all algorithms except for distributed Q-learning succeed. Indeed, an easy way to put Distributed Q-learning agents in the wrong is to insert stochastic rewards. So distributed Q-learning never manages the coordination in the partially stochastic Boutilier's game. When mis-coordination penalties are introduced in the deterministic game, decentralized Q-learning and WoLF-PHC fail because of shadowed equilibria. Distributed Q-learning, hysteretic Q-learning and SOoN succeed in the coordination. But in the stochastic version with $k = -100$, SOoN is the only one to be close to 100% of optimal joint action.

So SOoN gets best convergence results. In deterministic games, it turns towards the Distributed Q-learning. In the stochastic Boutilier's game without penalties, it turns towards the decentralized Q-learning. In the stochastic game with penalties, SOoN outperforms all other algorithms. *The linear interpolation to evaluate state-action couples is a pertinent heuristic to measure real values of actions.*

7.5.2 Two predators pursuit domains

The pursuit problem is a popular multi-agent domain in which some agents, called predators, have to capture one or several preys in a gridworld [48]. In the two agents pursuit domains, predators have to explicitly coordinate their actions in order to capture a single prey in a discrete 10×10 toroidal grid environment (Fig. 6a). At the beginning of each trial, predators and prey are set up in distinct random positions. At each time step, all predators simultaneously



(a) Complete 10×10 toroidal grid. (b) Possible capture position.

Figure 6: Two predators pursuit problem.

execute one of the 5 possible actions : move north, south, east, west or stand still. Moreover, noise is added to the environment through uncertainty in actions results. Any action moves a predator in the intended direction with probability 0.8, and otherwise a random action is applied. The prey moves according to a randomized policy : it remains on its current position with a probability of 0.2, and otherwise moves to ones of its free adjacent cells with uniform probability. A prey is captured when both predators are located in cells adjacent to the prey, and one of the two predators moves to the location of the prey while the other predator remains, for support, on its current position (Fig. 6b). A trials ends when the prey is captured.

The state space is represented by the relative position of the two predators to the prey. So each IL computes a Q_i table with $99 \times 98 \times 5$ values, *i.e.* 48510 different state-action pairs for each of both agents. A capture results in a reward $R = 10$ for every agent, and predators and prey are replaced to distinct random positions. Furthermore, predators receive penalties in case of mis-coordination. They both receive a penalty of -50 when one of them moves to the prey without support or when they end up in the same cell. Then both are moved to a random, empty cell. In all other situations, the reward is 0.

The main factors which complicate the coordination are action shadowing induced by penalties and the stochastic environment due to the random behavior of the prey and the uncertainty in actions results.

A trial consists of 10000 steps. We make out learning trials and greedy trials in which agents are following their learned greedy policies. After each learning trial is done a greedy trial. For each greedy trial, we plot the number of captures per trial and the sum of received rewards per agent during the trial. With the number of captures per trial, we can compare greedy policies learned with each algorithm; with the sum of received rewards, the number of collisions or non-support is observed. Results obtained on 10000 greedy trials are given in figure 7. We can notice that :

- greedy policies learned by decentralized Q-learning fail to coordinate,
- there is the same number of capture with WoLF-PHC and hysteretic Q-learning, but the second one creates less collisions,
- despite the stochastic environment, greedy policies learned by distributed Q-learning manage the coordination with around 45 captures per trial. However, collisions are not much

avoided,

- SOoN gets best results. After around 4500 trials, it outperforms distributed Q-learning. Two phases can be identified : during a first phase, SOoN behaves as distributed Q-learning (2000 first trials). Indeed, agents are initially optimistic. This first stage is in some way an adaptative phase to the environment; the frequency refines according to the stochasticity of the environment. Then evaluations swing between optimistic or neutral. The second phase is when SOoN manages the coordination : the number of captures increases and overcome all other results. So *SOoN realizes an automatic adaptation to the environment.*

7.5.3 Four agents pursuit domains

In the four agents pursuit domains, four predators have to coordinate their actions in order to surround a single prey in a discrete 7×7 toroidal grid environment (Fig. 8a). Agents observations suffer from low resolution. Indeed, a predator perceives the others according to the 8 cardinal directions and a close or distant criterion (Fig. 8b), so 16 perceptions. Given that each predator perceives its 3 teammates plus the prey, there are 16^4 possible observations per agent. Each agent has 5 possible actions, *i.e.* a Q_i-table of size $16^4 \times 5$ for each predator. Concerning the reinforcement function, if at least one agent is in the situation in figure 8c, every agent receives $R = 25$. When the prey is captured, they each receive $R = 100$. The low resolution in the perception induces stochasticity in the environment so the main factor here which complicates the coordination of predators is the noise.

A trial consists of 1000 steps. We make out learning trials and greedy trials in which agents are following their learned greedy policies. After each learning trial is done a greedy trial. For each greedy trial, we plot the number of captures per trial (Fig. 9). We can compare greedy policies learned with each algorithm. We can notice that :

- it is very difficult for distributed Q-learning to do the coordination. Indeed, optimistic agents capture the prey for the first time after 48000 trials, and they capture the prey 5 times per trial at 80000 trials. The low resolution in perceptions induces stochasticity in the environment which puts optimistic agents in the wrong,
- decentralized Q-learning shows interesting results in this stochastic environment. It confirms that this algorithm can be applied with success on some MAS and manages the coordination,
- WoLF-PHC is the first one to achieve captures but its performance increases slowly,
- hysteretic Q-learning manages first captures late, but its performance increases fast. It notably overcomes WoLF-PHC and is close to decentralized Q-learning after 80000 trials with around 60 captures per trial,
- SOoN gets best results. Two phases can be identified. During the first adaptative phase, agents are initially optimistic so SOoN does not manage any captures as distributed Q-learning. Then, once the frequency is fitted to the environment, the number of captures per trial increases. It is the second phase of coordination. After around 50000 trials, 90 captures are managed per trial. *SOoN realizes an automatic adaptation to the environment.*

We can also notice that greedy policies learned with SOoN and decentralized Q-learning are noised, unlike WoLF-PHC or distributed Q-learning policies. So even if SOoN realizes a fast and successful coordination, it is more sensitive to the stochasticity in this game.

7.5.4 Discrete smart surface with 270 agents

Given our real world application of a decentralized control of a smart surface, we have developed a discrete smart surface benchmark. The system is composed of 270 actuators organized in array (Fig. 10). At each time step, all actuators simultaneously execute one of the 5 possible actions : up, down, right, left or nop, represented by arrows and a circle. Uncertainty is added to actions results : with a probability 0.001, each actuator executes a random action. An object is situated on the surface. The motion of the object is proportional to the actions sum of the actuators which are under and by the object. The influence factor on the actions of the actuators by the object is three times higher than the influence factor on the actions of the actuators under the object. The object moves in the four cardinal directions but it does not rotate. Additionally, the object cannot get out of the surface and instead stays by the edge. The system's state is the index position of the object (according to its up and left corner). Each actuator is driven by an independent agent so it has only access to the index position and to its action. In real application, the full state observability can be done thanks to an overhead camera that extracts the index position.

The goal is for the agents to cooperate so as to move the object to a goal position. Then, all agents receive $R = 10$ and the trial ends. If the object is getting closer to the vertical edges of the surface, every agent receives a penalty ($R = -50$). Otherwise, the reward is 0.

A trial starts with the object at its initial position and ends when the object reaches the goal position (absorbing state). For each trial, we plot the number of steps to the goal and the sum of received rewards per agent (Fig. 11). With the sum of received rewards, we can observe if the object is getting closer to the vertical edges of the surface. With the number of steps to the goal, we can check if the learned policies are close to best policy achieved "by hand"‘ which makes 3 steps per trial (without uncertainty). We can notice that :

- the environment is stochastic so distributed Q-learning fails to coordinate. Moreover, optimistic agents are unaware of penalties so they do not avoid edges of the surface; the sum of received rewards per trial is low,
- decentralized Q-learning shows interesting results : after 300 trials, 20 steps are required to reach the goal and the object is getting closer to the vertical edges around 4 times per trial. However, learned policies are not stable. Mis-coordination penalties and the lack of robustness face to exploration lead to destruction of individual policies. During the experimentations, we notice that agents coordinate to avoid edges : the object move round in circles in the area without penalties,
- hysteretic Q-learning is close to decentralized Q-learning. But as these agents are chiefly optimistic, they do not avoid as much the edges and the sum of rewards increases slowly,
- with WoLF-PHC, learned policies are stable but slow,
- after 300 trials, best results are obtained with SOoN : 12 steps are required to reach the goal and the object is getting closer to the vertical edges around once per trial. Moreover, these performances are achieved from 100 trials. Again two phases draw attention. The first phase of adaptation when agents are initially optimistic and so fails to move the object to the goal. Then, SOoN adapts its frequency to the environment : the number of steps to goal decreases. So the second phase is the coordination.

7.6 Conclusion

In this section, the objective was to design an algorithm for ILs, robust face to exploration and mis-coordination factors in cooperative Markov games. Starting from the recursive FMQ, we show that using a one-step lookahead with the myopic frequency in multi-stage games could mislead the algorithm. So we introduced the computation of a farsighted frequency to detect the stochasticity of the game. This frequency is used in a heuristic to evaluate state-action couples in the Swing between Optimistic or Neutral algorithm. This heuristic is based on a linear interpolation between optimistic or neutral evaluations. Thus, SOoN realizes an automatic adaptation to the environment stochasticity of its evaluations. Especially, two phases are characteristic of this method. An adaptative phase when the frequency detects the stochasticity. Indeed, at the beginning, agents are optimistic. If the environment is deterministic and the agents manage the coordination, agents stay optimistic. Otherwise, the frequency adjusts evaluations between optimistic or neutral. When these evaluations are adjusted, the second phase is the successful coordination of agents. These both phases have been illustrated with some experimentations on Markov games. They confirm that the linear interpolation is a pertinent evaluation of real values of state-actions. They also highlight that SOoN is able to overcome mis-coordination factors and is robust face to exploration.

8 Conclusion

In this report, we focused on the learning of independent agents in cooperative multi-agent systems. First, we highlighted some of the difficulties encountered by ILs in this framework. Especially, we analyzed some factors which complicate the ILs coordination : shadowed equilibria, equilibria selection and noise. These factors have been identified and interpreted in matrix games. The impact of the exploration has also been studied as a major stake in the reinforcement learning of ILs.

Through a section suggesting a uniform notation, related algorithms based on Q-learning have been reviewed according to some factors : the robustness face to exploration, the learning of optimal individual policies despite shadowed equilibria and noise, and the selection of a single optimal equilibria. Thanks to this study, we focused on developing an algorithm robust face to the exploration strategy and able to overcome mis-coordination factors in cooperative Markov games. For that, we emphasized common points between FMQ heuristic and distributed Q-learning. On the one hand, distributed Q-learning is robust face to exploration and insured to converge toward a Pareto optimal Nash equilibrium in deterministic Markov games. On the other hand, FMQ outperforms the difficulty of weakly noisy matrix games by decoupling noise due to agents behaviors and noise due to the environment.

Thus, we contribute to improve the FMQ algorithm in matrix games and propose a recursive version which is easy-to-use, robust and able to solve hard repeated matrix games. We further proposer the SOoN algorithm, standing for “Swing between Optimistic or Neutral”. Thanks to the computation of a farsighted frequency and to a novel evaluation of the state-action values, this algorithm sways automatically from optimistic to neutral evaluation according to a detection of the noise in the environment. At the beginning, the environment is supposed to be deterministic and so agents are optimistic. Then, the SOoN automatically adapts its frequency to the stochasticity of the environment through the assessment of the probability of attaining the optimal gain.

We specify the choice of the two parameters of the SOoN so as to obtain an algorithm robust face to the exploration. The noise due to agents behaviors has been defined with the global exploration. We demonstrate empirically the robustness and advantages of the SOoN algorithm

Table 13: Characteristics of RL algorithms for independent learners in cooperative games. Entries marked with "NT" indicate it has not been tested.

	Matrix Games	Markov Games	Equilibria Selection	Shadowed Equilibria	Stochastic environment	Robust face to exploration
Decentralized Q-Learning [31]	✓	✓	✓ with GLIE		✓ partially	low
Distributed Q-Learning [38]	✓	✓	✓	✓		total
Lenient learners [43]	✓		NT	NT	NT	low
Hysteretic Q-Learning [42]	✓	✓	✓ with GLIE	✓	✓ partially	low
WoLF PHC [39]	✓	✓	✓		NT	good
FMQ [29]	✓		✓ with GLIE	✓	✓ partially	low
Recursive FMQ	✓		✓	✓	✓ partially	good
SOoN	✓	✓	✓	✓	✓ partially	good

on a number of multiple states cooperative Markov games with numerous agents. Results show that this algorithm overcomes all mis-coordination factors, even weakly noise Markov games, and is robust face to the exploration strategy (table 13). The SOoN manages the coordination thanks to two characteristic phases : an adaptative phase where the frequency automatically fits to the stochasticity, and the coordination phase where the agents coordinate themselves. Moreover, results confirm that the linear interpolation can be a pertinent evaluation of real values of state-action couples, especially in stochastic games.

Our research has raised a number of issues to be investigated in on-going and future work. The first perspective is to test the SOoN algorithm on a real problem as the decentralized control of a smart surface.

The second perspective is to improve the proposed evaluation of action. Indeed, we presented encouraging experimental results, but the proposed linear interpolation to evaluate a state-action couple could be modified. Specifically, a bound for the farsighted frequency could be defined to mark optimistic or neutral evaluation out.

An other prospect concerns a common issue in RL of making algorithms scale to large problems. Indeed, the SOoN algorithm uses explicit table representations of Q, Q_{max} and frequency functions. In larger state-action spaces application scenarios, explicit table representations become intractable. So it would be interesting to generalize and approximate these tables with some kind of function approximation mechanism. So an interesting direction of this work is to combine function approximation mechanism with the SOoN algorithm.

Appendix A :

The following were the learning and decay rates used for the results presented in §4. Here t is the number of repetitions of the game. With the detailed algorithmic descriptions in section 4 and these parameters details, all of the presented results are reproducible.

- Decentralized Q-learning

$$\alpha = 0.1 \quad \gamma = 0 \quad \tau = 5000 \times 0.997^t$$

- Distributed Q-learning

$$\alpha = 1 \quad \gamma = 0 \quad \epsilon = 0.05$$

- Hysteretic Q-learning

$$\alpha = 0.1 \quad \beta = 0.01 \quad \gamma = 0 \quad \tau = 5000e^{-0.003t}$$

- Original FMQ

$$\alpha = 0.1 \quad \gamma = 0 \quad c = 10 \quad \tau = 499e^{-0.006t} + 1$$

Appendix B :

In the Penalty game with $n > 2$ agents, each agent has 3 actions a, b and c. If half agents or more play a and the others play c, the received reward is 10. If less than half agents play a and the others play c, the received reward is -100 since they fail to coordinate. If half agents or more play b and the others play c, the received reward is 2. Otherwise, the reward is 0.

There are many Pareto optimal Nash equilibria when half agents or more play a and the others play c. These optimal equilibria are shadowed by penalties in case of mis-coordination. There are also many sub-optimal Nash equilibria when half agents or more play b and the others play c. So this game presents mis-coordination factors. Additionally, the reward can be noised : instead of receiving a reward equal to 2, 12 and 6 are received with equal probabilities.

The following were the learning and decay rates used for the results presented in §6.2. Here t is the number of repetitions of the game. With the detailed algorithmic descriptions and these parameters details, all of the presented results are reproducible.

- decentralized Q-learning

Table 14: Learning parameters choices with decentralized Q-learning.

agents number	repetitions per trial	algorithm parameters	decision method	exploration strategy
$n = 3$	$t = 30000$	$\alpha = 0.1$ $\gamma = 0$	softmax	GLIE $\tau(t) = 5000 * exp(-0.0003 * t)$
$n = 4$	$t = 50000$	$\alpha = 0.1$ $\gamma = 0$	softmax	GLIE $\tau(t) = 5000 * exp(-0.0002 * t)$
$n = 5$	$t = 80000$	$\alpha = 0.1$ $\gamma = 0$	softmax	GLIE $\tau(t) = 5000 * exp(-0.0001 * t)$

- recursive FMQ
- WoLF-PHC

Table 15: Learning parameters choices with recursive FMQ.

agents number	repetitions per trial	algorithm parameters	decision method	exploration strategy
$n = 3$	$t = 30000$	$\alpha = 0.1$ $\alpha_f = 0.01$ $\gamma = 0$ $Q_{max,ini} = -100$	ϵ-greedy	stationary $\psi = 0.1$
$n = 4$	$t = 50000$	$\alpha = 0.1$ $\alpha_f = 0.001$ $\gamma = 0$ $Q_{max,ini} = -100$	ϵ-greedy	stationary $\psi = 0.15$
$n = 5$	$t = 80000$	$\alpha = 0.1$ $\alpha_f = 0.0001$ $\gamma = 0$ $Q_{max,ini} = -100$	ϵ-greedy	stationary $\psi = 0.2$

Table 16: Learning parameters choices with WoLF-PHC.

agents number	repetitions per trial	algorithm parameters	decision method	exploration strategy
$n = 3$	$t = 30000$	$\alpha = 0.1$ $\delta_{lose} = 0.006$ $\gamma = 0$ $\delta_{win} = 0.001$	ϵ-greedy	stationary $\psi = 0.1$
$n = 4$	$t = 50000$	$\alpha = 0.1$ $\delta_{lose} = 0.0006$ $\gamma = 0$ $\delta_{win} = 0.0001$	ϵ-greedy	stationary $\psi = 0.15$
$n = 5$	$t = 80000$	$\alpha = 0.1$ $\delta_{lose} = 0.0006$ $\gamma = 0$ $\delta_{win} = 0.0001$	ϵ-greedy	stationary $\psi = 0.2$

Appendix C :

The following is the description of the cooperative Markov game in figure 3. Each agent has a choice of two actions a and b. The game starts in state s_1. The transitions on the figure are marked by a pair of corresponding actions, denoting agent 1's action and agent 2's action respectively. " * " is a wild card representing any action. So if both agents are coordinated on the joint action $< a, a >$ in the state s_k, they move to the next state s_{k+1}. If at least one agent plays b, they remain on the spot. When agents reach the absorbing state s_j, they receive a reward 10. All other states yield a reward of 0.

References

[1] Lucian Busoniu, Robert Babuska, and Bart De Schutter. Multi-agent reinforcement learning: A survey. In *Int. Conf. Control, Automation, Robotics and Vision*, pages 527–532, December 2006. 1

[2] Peter Stone and Manuela M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000. 1

[3] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, 1998. 1, 3

[4] Erfu Yang and Dongbing Gu. Multiagent reinforcement learning for multi-robot systems: A survey. Technical report, Department of Computer Science, University of Essex, 2004. 1

[5] C.H. Papadimitriou and J.N. Tsitsiklis. On the complexity of designing distributed protocols. *Information and Control*, 24(4):639–654, 1982. 1

[6] C.H. Papadimitriou and J.N. Tsitsiklis. The complexity of markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987. 1

[7] D. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of markov decision processes. *Math. Oper. Res.*, 27(4):819–840, 2002. 1

[8] Nancy Fulda and Dan Ventura. Predicting and preventing coordination problems in cooperative q-learning systems. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2007. 1, 6

[9] Gang Chen, Zhonghua Yang, Hao Ge, and Kiah Mok Goh. Coordinating multiple agents via reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 10(3):273–328, 2005. 1

[10] Andrew Garland and Richard Alterman. Autonomous agents that learn to better coordinate. *Autonomous Agents and Multi-Agent Systems*, 8(3):267–301, 2004. 1

[11] Robert H. Crites and Andrew G. Barto. Elevator group control using multiple reinforcement learning agents. *Machine Learning*, 33(2-3):235–262, 1998. 1

[12] Katja Verbeeck, Ann Nowé, Johan Parent, and Karl Tuyls. Exploring selfish reinforcement learning in repeated games with stochastic rewards. *Autonomous Agents and Multi-Agent Systems*, 14(3):239–269, 2007. 1

[13] Kagan Tumer and Adrian Agogino. Distributed agent-based air traffic flow management. In *AAMAS '07: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 1–8, New York, NY, USA, 2007. ACM. 1

[14] D.H. Wolpert, J. Sill, and K. Tumer. Reinforcement learning in distributed domains: Beyond team games. In *Proceedings of the Seventeenth IJCAI*, 2001. 1

[15] Y. Fukuta, Y.A. Chapuis, Y. Mita, and H. Fujita. Design, fabrication and control of mems-based actuator arrays for air-flow distributed micromanipulation. *J. Microelectromech. Syst.*, 15(4):912 – 926, 2006. 1

[16] Y. Uny Cao, Alex S. Fukunaga, and Andrew B. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4(1):7–23, March 1997. 1

[17] M. Tan. Multiagent reinforcement learning: Independent vs. cooperative agents. In *10th International Conference on Machine Learning*, pages 330–337, 1993. 2, 10

[18] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957. 2

[19] Richard Bellman. A markov decision process. *Journal of Mathematical Mechanics*, 6:679–684, 1957. 2

[20] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, 1994. 3

[21] L. P. Kaelbling, Michael Littman, and Andrew Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996. 3

[22] John Von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944. 3

[23] M. J. Osborne and A. Rubinstein. *A course in game theory*. MIT Press, 1994. 3, 4

[24] John F. Nash. Equilibrium points in n-person games. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 36, pages 48–49, 1950. 4

[25] L.S. Shapley. Stochastic games. *Proc. Nat. Acad. Sci. USA*, 39:1095–1100, 1953. 5

[26] Leonid Peshkin, Kee-Eung Kim, Nicolas Meuleau, and Leslie P. Kaelbling. Learning to cooperate via policy search. In *Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 307–314, San Francisco, CA, 2000. Morgan Kaufmann. 5

[27] N. Vlassis. A concise introduction to multiagent systems and distributed AI. Informatics Institute, University of Amsterdam, 2003. 6

[28] Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 746–752, 1998. 7, 10

[29] S. Kapetanakis and D. Kudenko. Reinforcement learning of coordination in cooperative multi-agent systems. In *Proceedings of the Nineteenth NCAI*, 2002. 7, 12, 15, 18, 34

[30] Craig Boutilier. Planning, learning and coordination in multiagent decision processes. In *Theoretical Aspects of Rationality and Knowledge*, pages 195–201, 1996. 8, 11

[31] C.J.C.H. Watkins and P. Dayan. Technical note: Q-learning. *Machine Learning*, 8:279–292, 1992. 9, 10, 15, 34

[32] Lucian Busoniu, Robert Babuska, and Bart De Schutter. Decentralized reinforcement learning control of a robotic manipulator. In *Proceedings of the 9th International Conference on Control, Automation, Robotics and Vision (ICARCV 2006)*, pages 1347–1352, Singapore, December 2006. 10

[33] Sandip Sen and Mahendra Sekaran. Individual learning of coordination knowledge. *JETAI*, 10(3):333–356, 1998. 10

[34] Sandip Sen, Mahendra Sekaran, and John Hale. Learning to coordinate without sharing information. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 426–431, Seattle, WA, 1994. 10

[35] Y. Wang and C. W. de Silva. Multi-robot box-pushing: Single-agent q-learning vs. team q-learning. In *Proc. of IROS*, pages 3694–3699, 2006. 10

[36] Satinder P. Singh, Tommi Jaakkola, Michael L. Littman, and Csaba Szepesvari. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, 38(3):287–308, 2000. 10

[37] S. Kapetanakis and D. Kudenko. Improving on the reinforcement learning of coordination in cooperative multi-agent systems. Second Symposium on Adaptive Agents and Multi-Agent Systems(AISB/AAMAS-II), Imperial College, London, April 2002. 10

[38] M. Lauer and M. Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proc. 17th International Conf. on Machine Learning*, pages 535–542. Morgan Kaufmann, San Francisco, CA, 2000. 11, 15, 18, 19, 22, 34

[39] Michael Bowling and Manuela Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136:215–250, 2002. 11, 15, 34

[40] Bikramjit Banerjee and Jing Peng. Adaptive policy gradient in multiagent learning. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 686–692, New York, NY, USA, 2003. ACM. 11

[41] M. Bowling. Convergence and no-regret in multiagent learning. In Y. Weiss L. K. Saul and L. Bottou, editors, *Advances in Neural Information Processing Systems*, pages 209–216. MIT Press, 2005. 11

[42] Laetitia Matignon, GuillaumeJ. Laurent, and Nadine Le Fort-Piat. Hysteretic q-learning :an algorithm for decentralized reinforcement learning in cooperative multi-agent teams. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems IROS 2007*, pages 64–69, San Diego, CA, USA, Nov. 2007. 11, 15, 34

[43] Liviu Panait, Keith Sullivan, and Sean Luke. Lenient learners in cooperative multiagent systems. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 801–803, New York, NY, USA, 2006. ACM Press. 12, 15, 19, 34

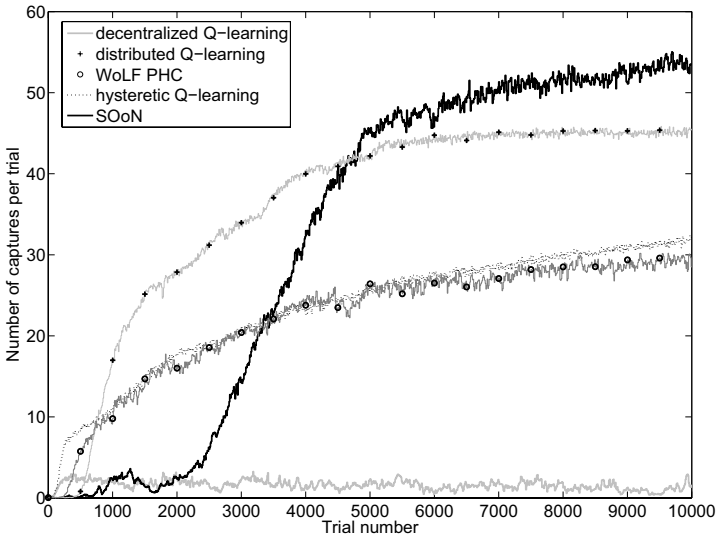
[44] Keith Sullivan, Liviu Panait, Gabriel Balan, and Sean Luke. Can good learners always compensate for poor learners? In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 804–806, New York, NY, USA, 2006. ACM Press. 12

[45] Spiros Kapetanakis and Daniel Kudenko. Reinforcement learning of coordination in heterogeneous cooperative multi-agent systems. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1258–1259, Washington, DC, USA, 2004. IEEE Computer Society. 12

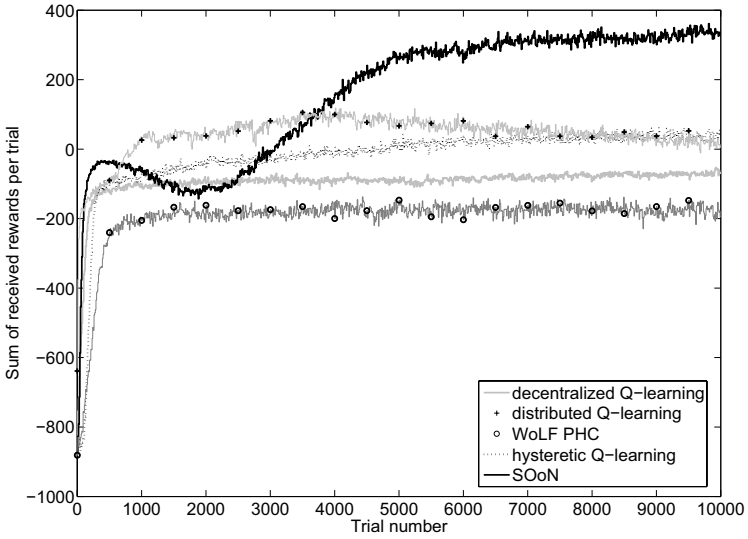
[46] Nicolas Meuleau and Paul Bourgine. Exploration of multi-state environments: Local measures and back-propagation of uncertainty. *Machine Learning*, 35(2):117–154, 1999. 26

[47] Craig Boutilier. Sequential optimality and coordination in multiagent systems. In *IJCAI*, pages 478–485, 1999. 28

[48] M. Benda, V. Jagannathan, and R. Dodhiawala. On optimal cooperation of knowledge sources - an experimental investigation. Technical Report BCS-G2010-280, Boeing Advanced Technology Center, Boeing Computing Services, Seattle,Washington, 1986. 29

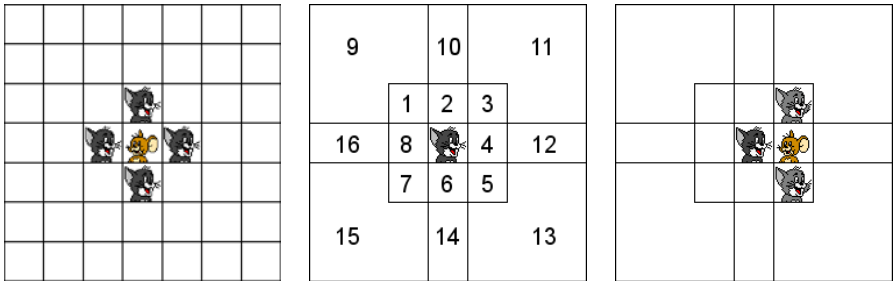


(a)



(b)

Figure 7: Number of captures and sum of received rewards per agent for 10000 steps (averaged over 20 runs) with $\alpha = 0.1$, $\beta = 0.01$, $\alpha_f = 0.01$, $\alpha_g = 0.3$, $\delta_{lose} = 0.06$, $\delta_{win} = 0.03$, $\gamma = 0.9$.



(a) Capture (b) Perceptions (c) Reinforcement

Figure 8: Four predators pursuit problem. a) The prey is captured. b) $(2 \times 8)^4$ perceptions per agent. c) The reinforcement is attributed in an individual way and is only function of local perceptions (and similar situations obtained by rotation of 90°).

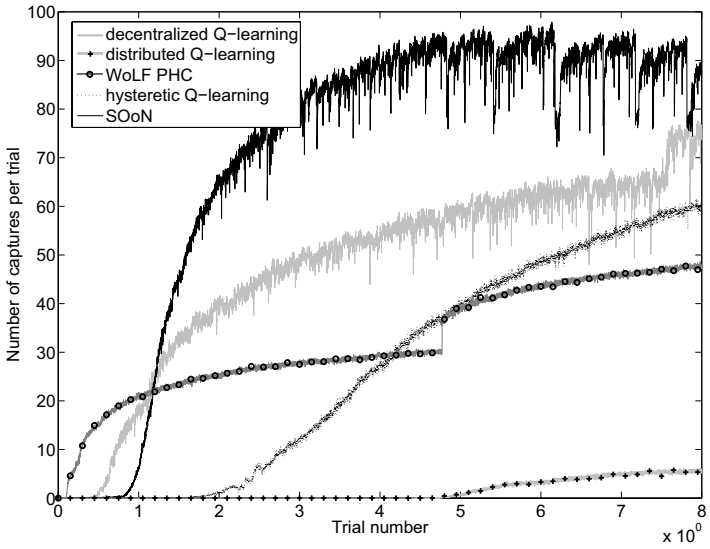


Figure 9: Number of captures for 80000 steps (averaged over 5 runs) with $\alpha = 0.3$, $\beta = 0.03$, $\alpha_f = 0.03$, $\alpha_g = 0.3$, $\delta_{lose} = 0.06$, $\delta_{win} = 0.03$, $\gamma = 0.9$.

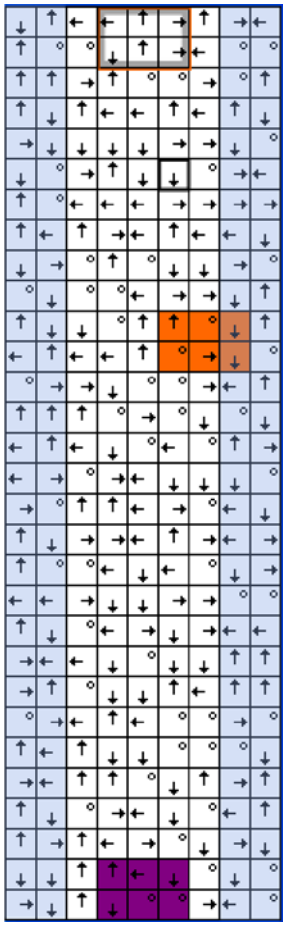


Figure 10: Discrete smart surface benchmark with 9×30 agents.

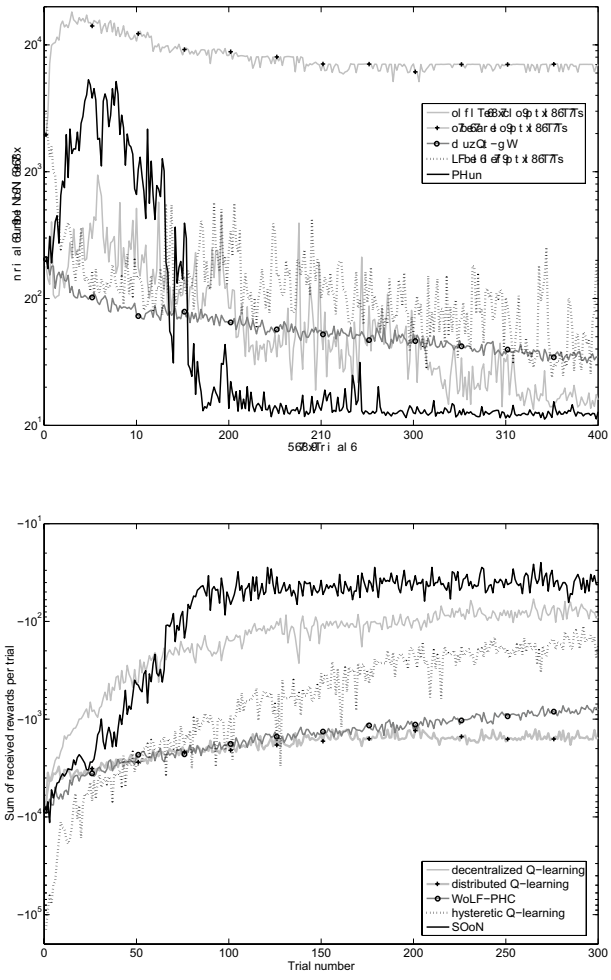


Figure 11: Steps to goal (on the top) and sum of rewards received at each trial by an agent (on the bottom) *vs.* trial number (averaged over 100 runs and logarithmic scale). $\alpha = 0.1$, $\alpha_f = 0.01$, $\alpha_g = 0.3$, $\gamma = 0.9$ and $\epsilon = 0.0005$.